UNIVERSIDADE FEDERAL DE SANTA CATARINA
PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

Projeto de uma camada de enlace de dados para um Sistema de Comunicação Tempo Real.

Dissertação submetida à Universidade Federal de Santa Catarina
como requisito parcial à obtenção do grau de

Mestre em Engenharia Elétrica

por

Georgina Vivanco

Florianópolis, 03 de junho de 1999

Projeto de uma camada de enlace de dados para um Sistema de Comunicação Real.

Georgina Vivanco

Esta dissertação foi julgada adequada para a obtenção do título de **Mestre em Engenharia** na especialidade **Engenharia Elétrica**, área de concentração **Controle, Automação e Informática Industrial**, e aprovada em sua forma final pelo curso de Pós-Graduação.

Florianópolis, 04 de junho de 1999.

Prof. Marcelo Ricardo Stemmer, Dr.
Orientador

Prof. Joni da Silva Fraga, Dr.
Co-orientador

Prof. Ildemar Cassana Decker, D.Sc.
Ccoordenador do Curso de Pós-Graduação em Engenharia Elétrica

Banca Examinadora

Prof. Marcelo Ricardo Stemmer, Dr.
Orientador, UFSC - DAS

Prof. Joni da Silva Fraga, Dr.
Co-orientador, UFSC - DAS

Prof. Jean Marie Farines, Dr. Ing
UFSC - DAS

Prof. Vitório Bruno Mazzola, Dr
UFSC - INE

Prof. Rômulo Silva de Oliveira, Dr.
UFRGS

À Manuel e Julia,

Agradecimentos

À meus orientadores Marcelo Ricardo Stemmer e Joni da Silva Fraga pela oportunidade brindada, pela confiança depositada e o trabalho de orientação no desenvolvimento do presente trabalho.

Aos membros da banca examinadora, que contribuíram opinando e sugerindo.

Ao CNPq e a UFSC pelo suporte material e financero, permitindo assim a realização deste trabalho.

Aos colegas Jose Miguel Eyzell Gonzalez, Cesar C. Torrico, Alejandro Garcia Ramirez, Eduardo Souza Machado da Silva, Felipe Luis Beck e Karina Acosta Barbosa pelas sugestões e esclarecimentos.

À Henrique Simas e ao Professor Augusto Coelho pelo apoio brindado na solução de problemas de infrestrutura.

À meus amigos Nelkis de la Orden Medina e Damian Rodriguez Sanchez pelo incentivo diante das dificuldades.

À Isa, Valdir, Valdira e a meu irmão Hugo pelo apoio e colaboração.

À meu companheiro Néstor e meus filhos Julia e Manuel por resistir.

Resumo

Este trabalho situa-se no âmbito dos sistemas de comunicação tempo real, ou seja, sistemas de comunicação que atendem os requisitos de natureza temporal na comunicação das mensagens, tendo que considerar com especial atenção a problemática do sistema de comunicação, devido à grande influência dele no possível cumprimento dos requerimentos temporais das mensagens.

Existem muitos trabalhos que tratam dita problematica com visões e propostas diferentes. Este trabalho apresenta as especificações de uma camada LLC para um sistema de comunicação tempo real. Entre os diferentes tipos de serviço fornecidos pela camada LLC, por motivos de simplicidade, especificou-se o serviço sem conexão e sem reconhecimento, mas que procura oferecer garantia de entrega de mensagens. No serviço sem conexão e sem reconhecimento, as abordagens de escalonamento são no sentido melhor esforço, mas como a nossa maior preocupação é a construção do teste de escalonabilidade modelando recursos concretos, para dominar a complexidade, a ideia é começar com serviços mais simples, com o objetivo de não inviabilizar as analises de escalonabilidade.

Se apresenta também uma análise de escalonabilidade que utiliza uma abordagem com uma política de atribuição de prioridades de forma dinâmica, o EDF("Earliest Deadline First"), que nos últimos tempos estão adquirindo maior importancia por suas características de maior eficiência no sentido de utilização do recurso.

Finalmente, a análise de escalonabilidade considera-se um modelo de mensagens que admite deadlines arbitrários, a existência de "jitter"e compartilhamento do recursos de comunicação. Como protocolo de implementação foi escolhido o protocolo determinista CSMA-DCR(Carrier Sense Access Method/Deterministic Collision Resolution).

Abstract

This work deals with real-time communication systems. These are communication systems that attend to time requirement, paying especial attention to the communication system problem itself due to the great influence it has in meeting the time requirement of the messages.

There are many work that treat this problem, following a variety of visions and proposals. This work presents the specifications of the LLC-layer for a real-time communication system. For simplicity reason, out of the different types of services offered by the LLC-layer, a connectionless and no-acknowledge service was specified, but with guaranteed messages delivery. In services without connection or acknowledge, the schedulability approaches are in the sense of the best effort. However, since the mean concern is to construct the schedulability test by modeling concrete resources, the idea is to start with simpler services letting the schedulability analysis remain feasible.

This work also present schedulability analysis based on the dynamic priority assignment policy, the EDF (Earliest Deadline First). This priority assignment policy have gained great important lately because of its grater efficiency in resourse utilization.

Finally, for the schedulability analysis, we consider a model for the messages that allows arbitrary deadlines, the existence of jitter and shared communication resources. As a protocol implementation we chose the deterministic CSMA-DCR (Carrier Sense Access Method/Deterministic Collision Resolution).

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

Neste capítulo se estabelece o contexto em que se desenvolve o presente trabalho, as motivações que o fazem possível surgir e o objetivo proposto. Em seguida apresentamos como está organizada a apresentação do texto.

Este trabalho situa-se no âmbito dos sistemas de comunicação tempo real. Denominam-se sistemas de comunicação tempo real aqueles sistemas que atendem os requisitos de natureza temporal na comunicação das mensagens, ou seja, onde os resultados, além de estar corretos do ponto de vista lógico, devem ser gerados no momento correto.

Dada a importância dos sistemas de comunicação tempo real na atualidade, existem muitos trabalhos com visões e propostas diferentes, onde a seleção de uma delas é uma solução de compromisso entre soluções mais simples mas também menos realistas, ou aumentar a complexidade das soluções em busca de hipóteses que se aproximem mais à realidade das aplicações.

Para facilitar o atendimento das restrições temporais das mensagens o sistema computacional deve ter um suporte adequado, com hardware e software que levem em conta tais restrições. Surge então a necessidade de uma arquitetura especial onde os parâmetros associados às restrições temporais sejam considerados. Também são necessários protocolos de acesso ao meio com tempo limitado e algoritmos de escalonamento adequados para atender às prioridades das mensagens.

1.1 A Motivação do Trabalho

Nos sistemas tempo real é importante poder determinar o comportamento temporal do sistema de comunicação envolvido, no LCMI se realizam esforços nesse sentido, sendo que o presente trabalho é uma continuação do trabalho [Fon97], no sentido de conseguir

resultados praticos e concretos envolvendo comunicação tempo real.

A motivação principal deste trabalho foi a possibilidade de definir um sistema de comunicação tempo real, onde se possa combinar um algoritmo de escalonamento como o proposto por Spuri em [Spu96], onde a atribuição das prioridades se realiza de forma dinâmica (EDF), com um protocolo determinista, como o CSMA-DCR(Carrier Sense Access Method/Deterministic Collision Resolution) proposto por Le Lann em [LR93], o qual apresenta características de bom desempenho, robustez e é auto adaptativo à carga do sistema.

Este seria um de uma série de trabalhos que especificariam os diferentes tipos de serviço fornecidos pela camada LLC, especificados no padrão ANSI/IEEE 802.3, utilizando o protocolo determinista CSMA-DCR.

1.2 Objetivo

O presente trabalho tem como objetivo apresentar as especificações de uma camada de enlace de dados para o envio de mensagens com restrições temporais, utilizando um serviço sem conexão e sem reconhecimento, com um protocolo determinista de acesso ao meio do tipo CSMA-DCR.

1.3 Organização do Trabalho

Este trabalho está dividido em seis capítulos:

- O capítulo 2 descreve os sistemas tempo real, apresentando sua definição, os requisitos que os caracterizam, assim como os diferentes critérios de classificação. Também são apresentados os critérios de classificação das mensagens de um sistema tempo real. Apresenta uma proposta de arquitetura tempo real, especificando sua estrutura funcional, protocolos, e os tipos de serviços fornecidos.

- O capítulo 3 apresenta a temática de escalonamento em sistemas tempo real. Define-se o modelo de escalonamento e as partes que o formam. Apresenta-se diferentes abordagens para realizar a análise de escalonabilidade e a justificativa da seleção de uma delas. Considera-se o custo de quantificação como conseqüência do limitado número de níveis de prioridade.

- O capítulo 4 apresenta uma visão mais detalhada da camada de acesso ao meio, utilizando o protocolo determinista CSMA-DCR, especificando suas características

temporais em termos de tempos máximo de latência em situações de instante crítico. Também apresenta-se a interface específica que implementa dito protocolo, assim como suas características fundamentais.

- O capítulo 5 apresenta uma proposta de camada LLC para um serviço sem conexão e sem reconhecimento para o envio de mensagens que exigem garantia. Finalmente se apresenta de forma detalhada o procedimento para realizar o teste de escalonabilidade, utilizando um algoritmo de escalonamento com atribuição dinâmica das prioridade (EDF) e um protocolo determinista de acesso ao meio, o CSMA-DCR.

- O capítulo 6 trata das conclusões do presente trabalho, assim como das perspectivas para futuros trabalhos.

Capítulo 2

Comunicação Tempo Real

2.1 Introdução

Os STR(Sistema Tempo Real) são identificados como aqueles sistemas submetidos a requisitos de natureza temporal. O requisito temporal é expresso usualmente através do "deadline", ou seja, o prazo máximo para a conclusão da tarefa.

Um STR [ARS91] é definido como um sistema cuja resposta correta ou válida depende não somente dos resultados lógicos de computação, senão também do tempo em que eles foram produzidos.

Outra forma de expressá-lo seria, os STRs se caracterizam pela resposta correta ou válida ("correctness") dentro dos prazos ("timeliness") impostos pelo ambiente. Além da correção lógica, existe a necessidade de correção temporal.

2.2 Requisitos de um Sistema Tempo Real

Existem vários requisitos que caracterizam um STR [ABB+92]:

- Requisitos de correção temporal (*timeliness*): o sistema deve reagir aos estímulos dentro de intervalos de tempo especificados (*deadline*).
- Requisitos de ordem (*ordeliness*): as relações de ordem entre as entradas (estímulos) do sistema devem se manter nas correspondentes saídas (respostas).
- Dados atualizados (*freshness*): o sistema deve fornecer ou processar dados o mais válido possível do ponto de vista do comportamento temporal requerido.
- Dados válidos (*correctness*): o sistema deve fornecer ou processar dados lógicos corretos.

- Previsibilidade do sistema (*predictability*): se refere à capacidade de conhecer o comportamento temporal de um sistema antes da sua execução, ou seja, é saber a priori, na etapa de projeto, se todas as tarefas serão executadas dentro de suas restrições temporais.

  Na literatura existem duas noções de previsibilidade, uma previsibilidade determinista que se refere a quando os deadlines de todas as tarefas são respeitados e outra previsibilidade probabilista baseada em estimativas que determinam a probabilidade dos deadlines serem respeitados.

  A previsibilidade aparece como um requisito fundamental para garantir as restrições temporais de um STR.

Existem diferentes critérios para classificar os STR. Em função dos requisitos, podem coexistir diferentes visões para estruturar uma classificação, dependendo de qual requisito se defina como o mais importante para considerar, por exemplo:

- a partir da *criticabilidade*; se nos basearmos nas conseqüências envolvidas com possíveis falhas no sistema, os STR podem ser classificados como:
  - STR crítico ("hard real-time system"): a ocorrência de falhas temporais ou um não atendimento no prazo pode resultar em conseqüências catastróficas para o sistema.
  - STR não crítico ("soft real-time system"): são aqueles sistemas que não são críticos, ou seja, aqueles em que as falhas são benignas.
- a partir dos *modos de demanda*; se nos basearmos na demanda de carga de processamento que um sistema deve atender, os sistemas podem ser classificados em:
  - sistemas com carga uniforme: quando a carga permanece estável, não varia.
  - sistemas com carga variável: quando a carga não é uniforme, varia desde um nível mínimo até um pico, as situações de pico de carga devem ser identificadas para ter condições de fazer uma estimativa dos recursos do sistema.
- a partir dos *modos de operação* ou respostas: se nos basearmos no grau de funcionalidade que é exigido do sistema para evitar catástrofes, existem dois modos de operação limites, a funcionalidade plena e parada(silêncio). Algumos sistemas tempo real somente aceitam a funcionalidade plena e em caso de falha, parada. Outros aceitam modos intermediários que podem fornecer desempenho e comportamentos degradados ou alternativos no caso de falhas e mesmo assim atenderem requisitos.

Um sistema tempo real centralizado, com um único computador pode ser representado como na figura 2.1. O sistema tempo real é um sistema computacional para o qual é requerida uma reação a estímulos(físicos ou lógicos) oriundos do ambiente, dentro de intervalos de tempo impostos pelo próprio ambiente.

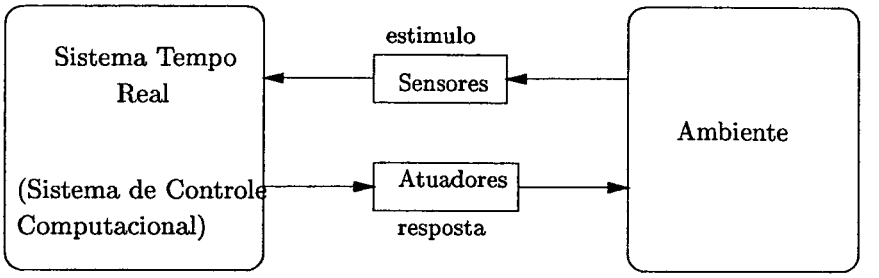

Figura 2.1: Sistema Tempo Real Centralizado

Nos últimos anos tem apresentado uma clara tendência para a distribuição das funções, adquirindo uma grande importancia os sistemas tempo real distribuídos, sistemas com vários computadores, como o que se mostra na figura 2.2.

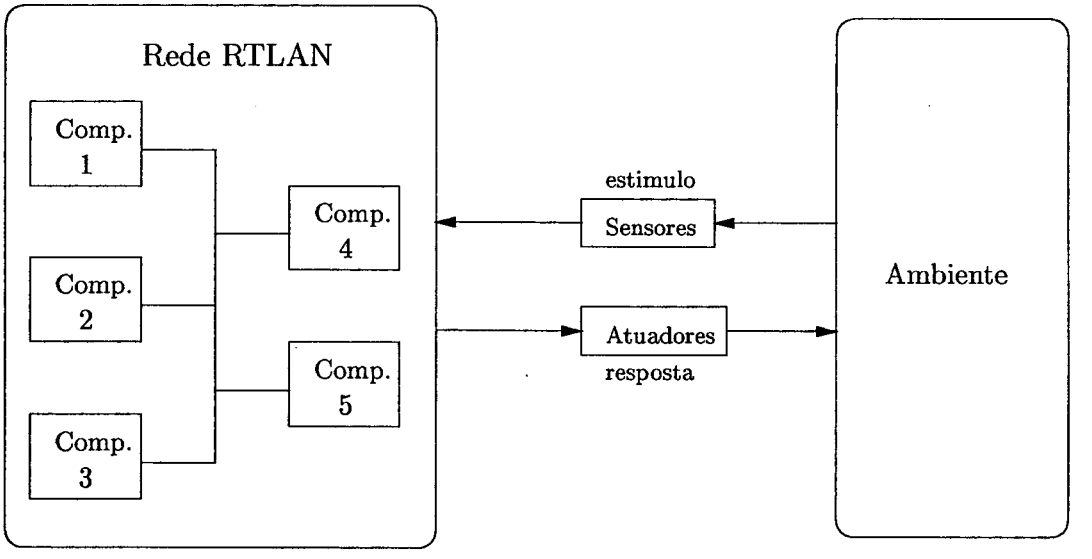

Figura 2.2: Sistema Tempo Real Distribuído

Com esta arquitetura distribuída, nos sistemas tempo real é importante poder determinar o comportamento temporal do sistema de comunicação envolvido, e é isto o que será abordado neste trabalho.

2.3 Mensagens em um Sistema de Comunicação Tempo Real

Denominaremos sistema de comunicação tempo real aquele sistema que atende os requisitos temporais de comunicação das mensagens.

Existem vários critérios [ARS91] para classificar as mensagens em um sistema de comunicação tempo real.

Segundo as suas exigências de garantia podem ser classificadas em:

- mensagens que exigem garantia ("guarantee seeking"): são as mensagens críticas ou essenciais para a operação correta do sistema. O sistema deve garantir que se a mensagem e a atividade referida pela mensagem foram aceitas, suas restrições temporais serão obedecidas.
- mensagens "best effort": são as mensagens que possuem restrições temporais, mas não precisam de uma garantia do sistema quanto a seu cumprimento. O sistema deve fazer o maior esforço para satisfazer as restrições de tempo, mas a perda ocasional do "deadline" não implica em grandes custos para o sistema.

Segundo a freqüência de ativações das mensagens, estas podem ser classificadas como:

- periódicas: tem que ser enviadas em intervalos conhecidos e fixos de tempo.
- aperiódicas: tem que ser enviadas a qualquer momento, sem período nem previsão.
- esporádicas: se caracterizam por um intervalo mínimo de tempo entre duas ativações consecutivas.

As mensagens são caracterizadas ou descritas através dos seguintes parâmetros:

- período **P**, para mensagem periódica.
- intervalo de tempo entre habilitações **I**, para mensagem esporádica.
- tempo de transmissão **C**.
- deadline **D**, prazo máximo para concluir a transmissão da mensagem.

Estes parâmetros para a caracterização das mensagens são usados em um dos problemas básicos dos sistemas de comunicação tempo real, a análise de escalonabilidade.

2.4 Arquitetura para Comunicação em Sistemas Tempo Real

Uma arquitetura de rede define um conjunto de serviços, protocolos e formatos de mensagens para a implementação dos serviços. As arquiteturas são tipicamente estruturadas em um conjunto de níveis funcionais, para modularizar e simplificar a implementação. Cada nível fornece serviços para o nível imediatamente superior, implementando os serviços através dos protocolos. Não é necessário para um nível superior o conhecimento dos detalhes da implementação do serviço no nível inferior.

Na continuação é apresentada uma arquitetura para redes locais para comunicações em STR distribuídos chamada de RTLAN(real-time local area network), permitindo às aplicações especificar seus requisitos temporais e fornecendo os mecanismos necessários para garantir ditos requisitos [ARS91].

2.5 Arquitetura RTLAN

Esta arquitetura de software é composta de quatro níveis ou camadas [ARS91]: o nível de aplicação, o nível controle lógico de enlace (LLC-Logical Link Control), o nível de controle de acesso ao meio (MAC-Medium Access Control) e o nível físico, como se ilustra na figura 2.3:

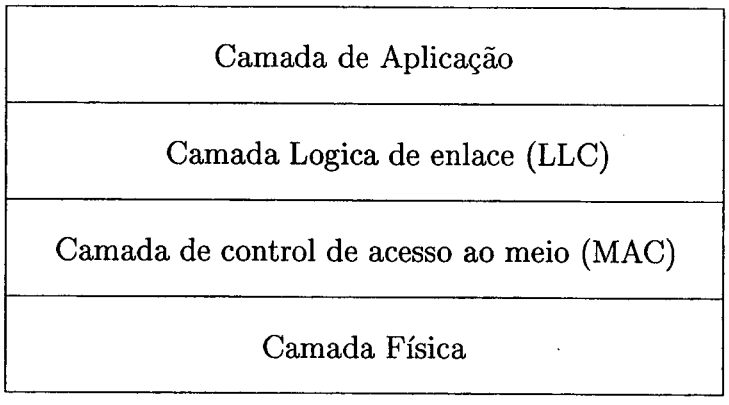

Figura 2.3: Arquitetura RTLAN

Esta proposta de arquitetura para aplicações tempo real adota a divisão no nível de enlace do padrão IEEE 802 em duas sub-camadas, a sub-camada de controle lógico de enlace e a sub-camada de controle de acesso ao meio. O objetivo desta divisão é permitir a definição de varias opções MAC, mantendo uma única interface, a sub-camada LLC.

Esta arquitetura apresenta as seguintes características [ARS91]:

- Aplicações tempo real: é uma arquitetura orientada às aplicações complexas de tempo real com restrições temporais.
- Serviços com restrições temporais: fornece serviços orientados à conexão e sem conexão, ambos considerando as restrições temporais das aplicações.
- Suporte de garantia no nível LLC: o suporte de garantia da camada LLC é bem mais complexo que numa arquitetura convencional devido à incorporação neste nível dos algoritmos de escalonamento, que, partindo das restrições temporais das mensagens, trata de garantir que seus requisitos serão respeitados.
- Protocolos MAC tempo real: a camada MAC utiliza protocolos especializados para tempo real, auxiliando à camada LLC no fornecimento dos serviços. Alguns protocolos são orientados para suportar a classe de serviço sem conexão enquanto outros para suportar a classe de serviço com conexão.
- Múltiplos canais físicos: a camada física consiste de múltiplos canais físicos e interfaces para tolerância a faltas, para garantir os requisitos funcionais e o desempenho.

2.6 Camada de Aplicação

Neste nível [SLC95] se definem as funções de gerenciamento e mecanismos genéricos que servem de suporte para aplicações distribuídas. Como é o nível que fornece serviços diretamente aos processos de aplicação, deve fornecer todos os serviços que podem ser usados por esses processos para trocar informações entre si.

Além da transferência de dados, inclui outros serviços como: sincronização das aplicações participantes, especificação de aspectos relativos a segurança como autenticação e controle de acesso.

2.7 Camada LLC

A camada LLC fornece serviços de comunicação à camada superior através da implementação de funções como gerenciamento da conexão, controle de erro e fluxo e fragmentação. Mas a diferença de uma arquitetura LAN convencional, a RTLAN fornece novas classes de serviço com um suporte para garantir as restrições temporais das mensagens.

2.7.1 Serviços

A camada LLC de uma RTLAN pode fornecer serviços orientados à conexão conhecidos como RTCOS("real-time connection-oriented service") e serviços sem conexão conhecidos como RTCLS("real-time connectionless service"). Estes serviços são acessíveis para a camada de aplicação através dos pontos de acesso ao serviço(SAP-Service Access Point) da camada LLC.

2.7.1.1 Serviço tempo real orientado a conexão

Um serviço tempo real orientado à conexão(RTCOS) é aquele que permite ao emissor especificar seus requisitos temporais no tempo de estabelecimento da conexão. RTCOS é um meio de suporte dos requisitos das mensagens que exigem garantia.

Caracteriza-se pelo estabelecimento de uma conexão lógica conhecida como conexão tempo real, a qual é estabelecida somente se os requisitos temporais das mensagens são garantidos; caso contrário, o emissor é informado que a conexão não pode ser estabelecida. Com o objetivo de estabelecer a conexão, o fornecedor do serviço RTCOS utiliza os serviços fornecidos pela camada MAC e os algoritmos de escalonamento.

Uma RTCOS estaria composta pelas seguintes operações:

- **rtcid < −− connect(receiverid, requirements)**

  O fornecedor do serviço de comunicação verifica se a conexão pode ser estabelecida garantindo os requisitos temporais especificados, nesse caso envia um identificador de conexão tempo real, caso contrário retorna um código de erro.

- **send(rtcid,message)**

  O emissor requisita a liberação de uma mensagem ao receptor final da conexão tempo real especificada.

- **message < −− receive(rtcid)**

  Recepção de uma mensagem enviada sobre a conexão tempo real especificada.

- **delete(rtcid)**

  O emissor ou receptor requisitam a finalização da conexão tempo real especificada.

2.7.1.2 Serviço sem conexão tempo real

Este serviço não envolve o estabelecimento de uma conexão. Em [ARS91], esta classe de serviços é apresentada para mensagens com restrições temporais "best effort", ou seja,

será feito o maior esforço em garantir as restrições temporais das mensagens.

As operações envolvidas neste tipo de serviço são:

- **send(receiver, message, requirements)**

  O emissor requisita a liberação de uma mensagem para um receptor, especificando os requisitos temporais (por exemplo o deadline).

- **(message, sender) < -- receive()**

  O receptor requisita a recepção de uma mensagem.

Para suportar o RTCLS, a camada LLC utiliza os serviços fornecidos pelos protocolos apropriados na camada MAC.

2.7.2 Fragmentação

Uma das funções implementadas pela camada LLC é a transformação das mensagens provenientes do nível de aplicação numa forma apropriada para a camada MAC. Para isto realiza-se a fragmentação, que é a divisão das mensagens longas em pacotes ou "frames"que satisfazem o máximo tamanho de pacote permitido pela camada MAC. Os requisitos temporais da mensagem original são propagados em todos os pacotes em que é fragmentada.

2.7.3 Garantia das conexões tempo real

Uma das características que distinguem uma RTLAN é o serviço orientado a conexão(RTCOS), que permite ao emissor especificar as restrições temporais no estabelecimento da conexão, e obter uma garantia do sistema que ditas restrições serão respeitadas.

Existem diferentes mecanismos ou abordagens que a camada LLC utiliza para obter tal garantia:

- **Atribuição de prioridades**

  Realiza-se uma atribuição de prioridades fixas a mensagens periódicas e estaticamente especificadas, e ao servidor periódico que atende a mensagens aperiódicas, assegurando que a soma de utilização das mensagens é limitada. Implementações com esta abordagem são o escalonamento utilizando a "taxa monotônica", o "deferrable server".

- **Circuito virtual**

  Esta abordagem pode ser utilizada para garantir os requisitos temporais tanto de mensagens estaticamente como dinamicamente conhecidas. Baseia-se na noção de circuito virtual tempo real, que é um canal lógico com a propriedade de que o tempo de serviço de um pacote, ou seja, o intervalo de tempo desde que o pacote entra em serviço até a sua transmissão completa está limitado pelo tamanho máximo de pacote. Este tamanho máximo de pacote é função do protocolo MAC utilizado.

- **Reserva de recursos**

  A abordagem do circuito virtual realiza uma distribuição da largura de banda do canal entre as diferentes estações, baseada no conhecimento local das requisições de conexão. Isto força a um cálculo pessimista assumindo o pior caso do tempo de serviço para cada pacote. Em contrapartida, a abordagem de reserva de recursos utiliza um conhecimento global de todas as requisições de conexão no sistema, incrementando portanto as chances de garantia na conexão.

  Esta abordagem trata o canal inteiro como uma única entidade de escalonamento, onde a camada LLC dos diferentes nós tentam escalonar suas requisições de conexão. Para isto é necessário que as camadas LLC de todos os nós possuam uma visão global do sistema, com todas as requisições reservadas aceitas de qualquer nó. Observa-se que como o algoritmo de escalonamento possui um conhecimento global de todos os pacotes de todos os nós que estão esperando para serem transmitidos, não é necessário assumir o pior caso de tempo de serviço de cada pacote. Isto aumenta as possibilidades de garantia, mas a implementação desta visão global requer uma complexidade maior na forma de mecanismos especiais para a tolerância a falhas, já que na presença delas, as cópias locais da fila global perdem consistência podendo resultar em possíveis acessos simultâneos ao canal por mais de um nó, violando desta forma a garantia oferecida.

2.7.4 Controle de fluxo e controle de erro

Esta são funções importantes pelas quais a camada LLC é responsável. Como controle de fluxo entende-se a técnica de sincronização que assegura que uma entidade emissora não cause um "overflow"de dados na entidade receptora. Normalmente o controle de fluxo é implementado através do reconhecimento baseado nos protocolos de janelas deslizantes("sliding window"). Como a utilização destes protocolos pode bloquear um emissor que espera o reconhecimento de um receptor, não é recomendável sua utilização em sis-

temas tempo real. Para garantir as restrições temporais dos pacotes, o número de "buffers" requeridos tem que ser reservados no nó destino. Esta reserva de "buffer" faz parte da negociação envolvida no estabelecimento da conexão tempo real. Para um serviço sem conexão o controle de fluxo pode ser exercido por meio da destruição ("dropping") dos pacotes que chegam quando o "buffer" está cheio.

Como controle de erro entende-se os mecanismos responsáveis pela detecção e correção de erro que transcorrem na transmissão de pacotes. Requer-se estes mecanismos porque é possível que um pacote seja perdido ou danificado pelos ruídos do canal de comunicação. O controle de erro é tipicamente implementado pelo reconhecimento positivo por parte do receptor de um pacote, quando ele chega sem ser danificado, e com reconhecimento negativo no caso contrário. No caso que o emissor não receba reconhecimento nenhum num período de tempo ou um reconhecimento negativo de um pacote, então ele retransmite o pacote ao receptor.

2.8 Camada MAC

Tipicamente uma rede local baseia-se no compartilhamento do canal físico, como um barramento ou um anel, chamado de canal de múltiplo acesso. Como múltiplos nós podem disputar simultaneamente o canal compartilhado, é necessário um mecanismo para arbitrar o acesso. Esta é a principal função da camada MAC.

Os protocolos de múltiplo acesso podem ser classificados em protocolos tempo real e não tempo real dependendo de fornecerem ou não suporte para comunicação tempo real.

Tomando como base as abordagens para a camada LLC que fornecem garantia apresentada anteriormente (atribuição de prioridades, circuito virtual tempo real e reserva de recursos), apresenta-se na continuação os protocolos MAC que implementam os serviços requeridos pela camada LLC para cada uma das abordagens.

- Atribuição de prioridades: requer protocolos de resolução de prioridades no nível MAC, seleciona-se para transmitir o pacote com maior prioridade de todos os pacotes contidos no sistema. Na continuação se apresentarão alguns em função da topologia da rede local.

  - Para redes locais com uma topologia em anel podemos mencionar o token ring standard(IEEE 802.5), onde o acesso ao canal é arbitrado através de um pequeno campo de controle conhecido como token. Somente a estação que possui o token pode transmitir, depois que finaliza a transmissão ou se não tem pacote para transmitir o token é passado para a próxima estação. A

resolução de prioridades é resolvida através de dois campos presentes no token, eles são o campo de prioridade e o campo de reserva.

- Para redes locais com topologia em barramento podem ser apresentadas três categorias:
  - * "deference delays": no final da transmissão de cada pacote, cada nó adia a transmissão de seu pacote por um tempo associado à diferença de $p_{max} - p$, onde p é a prioridade associada ao nó e p_{max} a máxima prioridade possível, depois de transcorrida esta diferença de tempo o nó escuta o canal, se está ocioso quer dizer que não tem outro nó com prioridade maior que a dele, estando autorizado a transmitir.
  - * "preamble lengths": no final da transmissão de cada pacote, cada nó transmite um preâmbulo com um tamanho associado a $p - p_{min}$, onde p_{min} é a menor prioridade possível, então, se a prioridade é maior, maior é o preâmbulo. A colisão se produz no período de tempo de transmissão dos preâmbulos, somente o nó que detecta o canal ocioso no final da transmissão de seu preâmbulo está autorizado a transmitir, porque é ele quem apresenta a maior prioridade.
  - * "forcing headers": este esquema caracteriza-se pela transmissão de um cabeçalho antes da transmissão do pacote, ou seja, todos os nós com informação para transmitir, transmitem sua prioridade desde o bit mais significativo até o menos significativo de forma simultânea em slots consecutivos, escutando o canal enquanto se faz dita transmissão, cada nó faz um OR entre o bit que ele transmite e o bit que recebe, se por exemplo ele envia um bit 0 no slot e recebe um bit 1, significa que existe uma estação com prioridade maior que está disputando o canal, a estação com o bit 0 fica fora da disputa, no final deste período de cabeçalho somente a estação com maior prioridade resta, fazendo a transmissão de seu pacote até completar. O tamanho deste cabeçalho depende do número de bits requeridos para representar o intervalo $[p_{min}, p_{max}]$. Exemplo é o CSMA/CA.

- Circuito virtual tempo real: requer protocolos que possam garantir um tempo de serviço limitado, este tempo de serviço está composto por dois elementos: o tempo de acesso ao canal físico e o tempo de transmissão do pacote. Exemplos de protocolos com este comportamento são:
  - TDMA: (Time Division Multiple Access), onde o tempo é dividido em tama-

nhos fixos conhecidos como "frames"e cada "frame"é subdividido em slots, com pelo menos um slot por nó.

- Outro exemplo são os protocolos "token-passing", onde os nós estão organizados em anéis lógicos ou físicos, e um token real ou virtual circula no anel outorgando a permissão de transmissão a cada nó por um tempo limitado.
- O CSMA-DCR é outro exemplo deste tipo de protocolos, porque substitui a espera aleatória de acesso ao canal do CSMA/CD por uma espera limitada. Este protocolo será especificado com mais detalhe no capítulo 4.
- Finalmente temos os "waiting room protocol"que estão baseados no quarto lógico de espera no qual o pacote deve entrar antes de ser transmitido. Um pacote pode entrar somente se o quarto está vazio, podendo acontecer que vários pacotes provenientes de diferentes nós encontrem o quarto vazio, tendo que estabelecer alguma ordem para transmitir ditos pacotes.

- Reserva de recursos: requer protocolos de reserva, para isto é necessário uma visão global de todas as requisições reservadas aceitas feitas em todos os nós do sistema. O protocolo PODA(Priority Oriented Demand Assignment) é um exemplo desta abordagem. O tempo do canal é dividido em "frames", cada um dos quais consiste de um subframe de informação e de um subframe de controle. O subframe de informação é utilizado para a transmissão dos pacotes escalonados e o subframe de controle é utilizado para o "broadcast"de reserva. Neste protocolo todos os nós devem ter uma cópia local da fila global de escalonamento. Alocando o tempo de canal à estações de acordo a esta fila. O principal problema deste protocolo é a falta de robustez na presença de erro, para o qual é necessário implementar mecanismos especiais para a tolerância a falhas.

Para um serviço tempo real sem conexão implementa-se protocolos do tipo "best effort", os quais consideram as restrições temporais dos pacotes, e no entanto não garantem seu cumprimento, tentam diminuir ou minimizar o número de mensagens em que não são respeitados seus deadlines.

2.9 Camada Física

A função do nível físico é permitir o envio de uma cadeia de bits pela rede sem se preocupar com o seu significado ou com a forma como esses bits são agrupados. Não é função desse nível tratar de problemas tais como erros de transmissão [SLC95].

Este nível fornece as características mecânicas, elétricas, funcionais e de procedimento para ativar, manter e desativar as conexões físicas para a transmissão de bits entre entidades de nível de enlace.

2.10 A arquitetura proposta

A arquitetura adotada neste trabalho para sistemas tempo real, proposta em [ARS91], apresenta como principal contribuição o esquema proposto para a escolha da combinação de protocolos MAC e LLC a partir da sua classificação pelo tipo de abordagem utilizada para a garantia, mas o critério que usa para a escolha de serviços que garantem a entrega das mensagens também é uma limitação desta arquitetura, pois as mensagens que exigem garantia estão associadas somente a um serviço orientado a conexão, enquanto as mensagens "best effort"estão associadas a um serviço sem conexão.

Com o propósito de buscar uma maior flexibilidade, no sentido de não impor um tipo de serviço específico em função do tipo de mensagens que está-se querendo transmitir, é que a proposta de arquitetura para o sistema apresentado esta baseado na proposta de [ARS91], mas que permita transmitir mensagens do tipo "guarantee seeking"utilizando um serviço sem conexão e sem reconhecimento.

A garantia de atendimento dos requisitos temporais nos sistemas tempo real depende da abordagem adotada para a garantia na execução dos serviços, assim como da combinação dos protocolos oferecidos pelas diversas camadas. Os protocolos de implementação devem possuir características deterministas, porque a base da garantia está no teste que se realiza na fase de projeto, onde são considerados todos os aspectos temporais envolvidos no sistema de comunicação.

2.11 Conclusões

Neste capítulo foram apresentados os conceitos básicos envolvendo STR e particularmente a problemática de comunicação em tempo real.

Existem diferentes visões para classificar um sistema tempo real dependentes do critério utilizado para sua caracterização. O presente trabalho parte da criticabilidade, e utiliza este critério como o mais importante para localizar o sistema que está-se especificando. O presente trabalho aborda a problemática de comunicação em sistemas tempo real crítico ("hard real-time systems"), onde os requisitos temporais devem ser garantidos, caso contrário pode resultar em conseqüências catastróficas para o sistema.

Outro aspecto abordado neste capítulo é o tipo de mensagens utilizadas num sistema tempo real e as possíveis classificações delas. Como o sistema de comunicação que está sendo especificado se destina à aplicações em sistemas tempo real críticos, as mensagens utilizadas na comunicação são do tipo que exigem garantia("guarantee seeking"). O sistema deve garantir que, se a mensagem e a atividade referida pela mensagem foram aceitas, suas restrições temporais serão obedecidas.

Se estabelece a arquitetura adotada para um STR, especificando os serviços fornecidos, detalhando os protocolos que implementam ditos serviços, mas que garantem a entrega de mensagens dentro de seus deadlines.

A continuação abordaremos um problema básico nos sistemas de comunicação tempo real, o escalonamento de mensagens.

Capítulo 3

Escalonamento de mensagens em Sistemas Tempo Real

3.1 Introdução

Neste capítulo se aborda o escalonamento em sistemas tempo real, apresentando uma classificação dos sistemas tempo real. Logo define-se o modelo de escalonamento e seus componentes. Apresenta diferentes abordagens para realizar a análise de escalonabilidade.

3.2 Escalonamento em Sistemas Tempo Real

Existem diferentes abordagens para a classificação dos sistemas tempo real, mas existem alguns aspectos relevantes para qualquer tentativa de classificação como: a previsibilidade, utilização dos recursos e algoritmos de escalonamentos utilizados.

Na literatura são identificadas diferentes abordagens de escalonamento tempo real. Na figura 3.1 apresentamos dita classificação, onde observa-se como a previsibilidade é determinante, ou seja, se o que se busca é um sistema tempo real com uma previsibilidade determinista ou probabilista. As propriedades do modelo de tarefas devem suportar a abordagem escolhida, assim como os algoritmos de escalonamento devem ser compatíveis com o modelo de tarefas adotado e os objetivos da abordagem escolhida.

Dentro das diferentes abordagens de escalonamento em sistemas tempo real, este trabalho estaria classificado entre os sistemas com garantia em tempo de projeto, e dentro deste grupo nos baseados em teste de escalonabilidade e prioridades. Ou seja, existe um teste de escalonabilidade em tempo de projeto, que considerando o instante crítico determina se existe garantia de que todas às mensagens serão transmitidas dentro de seus

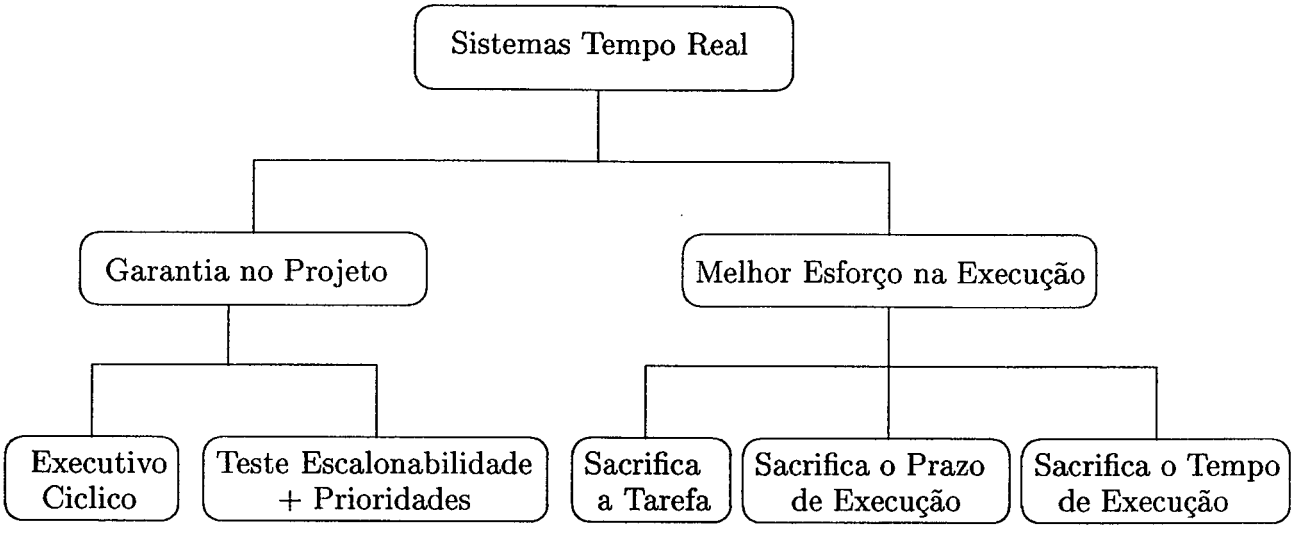

Figura 3.1: Abordagens para Sistemas Tempo Real

deadlines e um escalonador dinâmico baseado em prioridades que libera as mensagens em tempo de execução. A atribuição de prioridades as mensagens se faz de maneira dinâmica.

Esta garantia é obtida a partir de uma determinada hipótese de carga. As condições normalmente aceitas para tal garantia envolvem [Oli97]:

- que a carga deve ser estática, ou seja, limitada e conhecida em tempo de projeto
- deve-se reservar recursos para a execução de todas as tarefas no pior caso

Existem muitos trabalhos dentro desta abordagem(garantia em tempo de projeto) que se diferenciam no critério de atribuição das prioridades ao modelo de tarefas adotado e nos testes de escalonabilidade envolvidos.

A atribuição de prioridades pode ser feita de maneira estática, ou seja, à cada tarefa atribui-se uma prioridade de forma estática em tempo de projeto, em função de algum parâmetro da tarefa(período ou deadline), ou pode ser feita de maneira dinâmica, ou seja, em tempo de execução, em função da relação de algum parâmetro(deadline ou folga) com o tempo corrente.

Os testes de escalonabilidade historicamente seguem duas abordagens diferentes [Fid98]: os testes baseados na noção de utilização do processador, ou seja, a porcentagem de tempo de processador que pode ser ocupado por tarefas mais prioritárias e os testes baseados em tempos de respostas, ou seja, a exata duração considerando que as tarefas podem ser demoradas.

3.3 Modelo de Escalonamento

O escalonamento de mensagens surge da necessidade de compartilhar recursos (meio de comunicação, etc) entre diferentes usuários. O objetivo é definir o quê queremos escalonar, onde, e como. Para responder a estas três perguntas define-se um modelo de escalonamento baseado em modelo de mensagens, que descreve as mensagens a serem escalonadas com suas restrições temporais, o que responde à primeira pergunta, modela-se o recurso compartilhado o qual responde a segunda, e por último define-se uma política de escalonamento que satisfaz as restrições impostas ao conjunto.

O modelo de escalonamento estaria composto então de:

- *Modelo de mensagens:* representa uma caracterização das mensagens envolvidas no trafego entre os nós da rede, consideramos estas mensagens periódicas e esporádicas, representando uma carga estática conhecida a priori, sendo possível realizar um escalonamento com garantia em tempo de projeto.

  As mensagens são caracterizadas pela tripla C, T, D.

  - C representa o tempo máximo de transmissão da mensagem, este tempo é função do comprimento da mensagem e da taxa de transmissão.
  - T é o período para as mensagens periódicas e o intervalo mínimo de chegada de mensagens consecutivas para as esporádicas.
  - D representa a restrição temporal da mensagem, dada pelo deadline, é o tempo limite para que se complete a transmissão da mensagem entre os nós.

- *modelo do recurso:* representa as propriedades temporais do sistema de comunicação e como estas características influenciam no atraso fim-a-fim na transmissão de mensagens. Dentro das características do sistema de comunicação poderíamos citar algumas como número de estações que estão interligadas, topologia da rede utilizada, tipo de protocolo MAC usado, características físicas da rede, taxa de transmissão das mensagens, velocidade de propagação no meio físico. Estas características determinam os valores de parâmetros associados ao recurso como: sobrecargas das mensagens produto da fragmentação das mensagens e seu encapsulamento, sobrecargas do sistema produto da transmissão de quadros de controle, bloqueio por inversão de prioridades e outras.

- *Política de escalonamento:* como política de escalonamento se refere ao critério de decisão na ordenação de mensagens. Como tratamos de uma abordagem por prio-

ridades, as mensagens são ordenadas, no caso, segundo suas prioridades. Existem duas políticas de escalonamento na atribuição das prioridades:

- política de prioridades fixas, onde a prioridade está associada a certo parâmetro(período, deadline) da tarefa, fixando de forma estática a prioridade da tarefa, por exemplo, no Taxa Monotônica ("Rate Monotonic") a tarefa com menor período tem atribuída maior prioridade, no Deadline Monotônico("Deadline Monotonic") a tarefa com menor deadline tem atribuída a prioridade mais alta.
- políticas de escalonamento com prioridades dinâmicas como são o Próximo Deadline("Earliest Deadline First") e o Menor Folga("Least Laxity First"). Na primeira, à tarefa com deadline mais próximo do tempo corrente é atribuída a maior prioridade. Na segunda, à tarefa com menor folga é atribuída a mais alta prioridade.

3.4 Diferentes testes para a análise de escalonabilidade

A continuação apresentaremos diferentes testes para a análise de escalonabilidade. Dois desses testes utilizam prioridades fixas e um prioridades dinâmicas.

3.4.1 Teste para a análise de escalonabilidade baseada em interferências [TBW94]

Esta técnica baseia-se no calculo do *pior caso de tempo de resposta* de uma mensagem. Entendendo por *pior caso no tempo de resposta* como o tempo que transcorre entre a chegada da mensagem e o término da sua transmissão. Para isto utiliza-se o conceito de "período de ocupação" ("busy period") para prioridades fixas e um escalonador preemptivo em tempo de execução. Neste trabalho se consideram mensagens periódicas ou esporádicas com deadlines arbitrários. As mensagens podem sofrer de "release jitter" na liberação. As ocorrências de mensagens são identificadas e tratadas como ocorrências de tarefas no modelo de escalonamento.

A continuação apresentamos algumas considerações necessárias para este teste:

- que a atribuição das prioridades das mensagens é realizada de forma estática.

- que as mensagens são caracterizadas pelo pior caso de tempo entre ativações, chamado de período T.
- que cada mensagem possui um tempo máximo de transmissão C.
- o instante crítico, ou seja, quando todas as mensagens estão prontas para ser liberadas no mesmo instante de tempo.
- a existência de interferência de mensagens mais prioritárias que a mensagem i.
- deadlines arbitrários, ou seja que o deadline da mensagem pode ser maior, menor ou igual ao seu período.

A abordagem utiliza a noção de "busy period"definido como o máximo tempo que o suporte de comunicação está sendo ocupado por mensagens com prioridades maior o igual à prioridade da mensagem i. O pior caso no tempo de resposta pode ser encontrado examinando uma série de janelas, considerando que as janelas começam com a chegada de uma mensagem m_i na fila de transmissão e termina quando a transmissão da mensagem m_i seja completada no nó de origem. Como considera-se que as mensagens podem ter deadlines arbitrários, elas podem sofrer de *interferência interna*, ou seja que os q pedidos anteriores de transmissão de uma mensagem m_i podem ser acumulados na fila de transmissão. A partir da análise das diferentes janelas obtidas, pode-se dizer que o *pior caso no tempo de resposta* (r_i) de uma mensagem i é:

$$r_i = max_{q=0,1,2}(w_i(q) - qT_i) \tag{3.1}$$

onde a largura da janela $w_i(q)$ e dado por:

$$w_i(q) = (q+1)C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i(q)}{T_j} \right\rceil C_j \tag{3.2}$$

Na expressão acima $hp(i)$ representa o conjunto de mensagens com prioridades mais altas que i e o termo $\left\lceil \frac{w_i}{T_j} \right\rceil C_j$ é o tempo de interferência que a mensagem i pode sofrer de uma mensagem j de maior prioridade.

A mensagem i terá suas restrições temporais atendidas se a largura de tempo r_i for menor que o seu deadline. Como w_i aparece nos dois lados da equação, é necessário a introdução de um método iterativo para encontrar seu valor. A condição para a equação iterativa convergir a um valor, em um número finito de passos é que a utilização do sistema dos i níveis mais altos de prioridade seja menor que a unidade.

O valor de $w_i(q)$ pode ser calculado por iteração como:

$$w_i(q)^0 = (q+1)\, C_i$$

e se observamos que,

$$w_i(q+1) \geq w_i(q)$$

então,

$$w_i(q)^0 = w_i(q-1)$$

obtendo a convergência com,

$$w_i(0)^0 = C_i$$

O conjunto de tarefas é escalonável se,

$$\forall i \quad r_i \leq D_i$$

Observando a figura 3.2, a terceira ativação da mensagem i (q=2) finaliza em um tempo $w = 2C_{high} + 3C_i$, este é o valor de w ao qual a equação 3.2 converge quando $q = 2$. Esta ativação é liberada no tempo $2T_i$, por tanto o tempo de resposta desta ativação é $w = 2C_{high} + 3C_i - 2T_i$. Avaliando os tempos de respostas das cinco ativações que começam no "busy period"pode-se observar que a terceira é a representa o pior caso no tempo de resposta.

A continuação detalharemos três termos que devem ser considerados para estender o teste de escalonabilidade para um modelo mais realista, eles são o tempo de bloqueio por inversão de prioridade, a existência de "release jitter"e o atraso devido a propagação elétrica no meio físico.

- Como **máximo tempo de bloqueio** (B_i) entende-se o tempo que transcorre desde que uma mensagem de prioridade i é atrasada na sua transmissão causada pela transmissão em curso de uma mensagem de menor prioridade.

  De ser necessário que uma mensagem seja fragmentada em pacotes, podemos assumir que todos os pacotes serão enfileirados no mesmo tempo, e uma vez que o primeiro pacote ganha o direito a transmitir no suporte de comunicação, não é possível que outra mensagem de menor prioridade ganhe o direito, portanto, em um "busy period"nível i pode acontecer no máximo um bloqueio por inversão de prioridades e como a janela $w_i(q)$ representa a execução de um "busy period"nível i, que começa no tempo qT_i anterior à ativação de i que estamos analisando, temos então

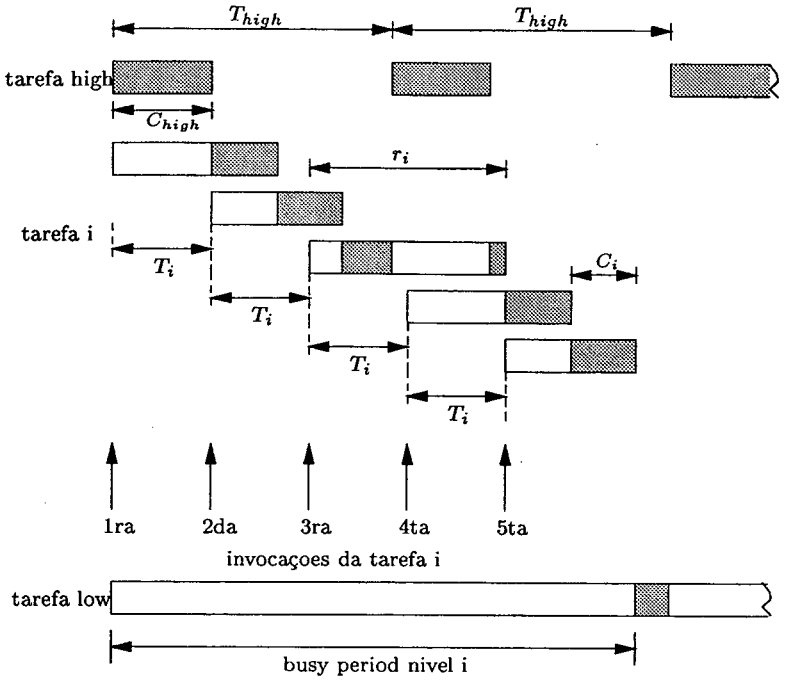

Figura 3.2: Busy period nível i

que considerar a existência de dito bloqueio, adicionando o termo B_i na expressão de w_i.

- o **"release jitter"**, ou seja, as tarefas podem sofrer um tempo de demora entre o tempo de chegada da tarefa e sua liberação. O "release jitter"não afeta o tempo de computação de uma tarefa, mas sim afeta o seu tempo de resposta(r_i).
- o atraso devido a **propagação elétrica** no meio físico($\wp$).

Incorporando então estes termos, o pior tempo de resposta de uma mensagem i ficaria como:

$$r_i = max_{q=0,1,2}(w_i(q) + J_i - qT_i) \tag{3.3}$$

onde $w_i(q)$ e dado por:

$$w_i(q) = (q+1)C_i + \sum_{\forall j \in hp(i)} \left\lceil \frac{w_i(q) + J_j}{T_j} \right\rceil C_j + B + \gamma \tag{3.4}$$

3.4.2 Teste para a análise de escalonabilidade baseada em saturação [KS94], [KSS95b]

Este teste é proposto para modelos de escalonamentos para sistemas preemptivos com prioridades fixas. Para isto baseia-se no conceito de trabalho cumulativo $L_i(t)$, ou seja, o cálculo do tempo de utilização do recurso compartilhado, neste caso o suporte de comunicação, na transmissão das mensagens com prioridades maiores ou iguais à prioridade da mensagem analisada durante a janela de tempo t.

Considerando o instante crítico e que as mensagens são periódicas com deadlines menores ou iguais aos respectivos períodos, o trabalho cumulativo para uma mensagem de prioridade i é dado por:

$$L_i(t) = \sum_{j=i}^{i} C_j \left\lceil \frac{t}{T_j} \right\rceil \tag{3.5}$$

Para que exista garantia de entregue de uma mensagem m_i, o trabalho cumulativo envolvendo as mensagens com maior ou igual prioridade ao nível i deve ser menor que a janela de tempo disponível para sua transmissão. Portanto o teste de escalonabilidade de um conjunto de mensagens com prioridades i variando de 1 a n, necessário e suficiente, é dado pela seguinte condição:

$$\forall i, 1 \leq i \leq n, \quad D_i \leq T_i \qquad min_{0 \leq t \leq D_i} \frac{L_i(t)}{t} \leq 1 \tag{3.6}$$

Utilizando a expressão de $L_i(t)$, a condição acima pode ser re-escrita como:

$$\forall i, 1 \leq i \leq n, \quad D_i \leq T_i \qquad min_{0 \leq t \leq D_i} \sum_{j=i}^{i} \frac{C_j}{t} \left\lceil \frac{t}{T_j} \right\rceil \leq 1 \tag{3.7}$$

A condição de teste anterior pode ser considerada para um modelo de escalonamento ideal, porque não considera os custos de preempção de mensagens e sobrecargas produto da implementação dos serviços no sistema.

Em [KSS95b] são apresentadas as extensões destas condições para um modelo real. Na figura 3.3 se mostra os componentes requeridos para a criação de uma metodologia de modelagem genérica dos recursos da rede. Existindo um modelo de escalonamento para cada política de escalonamento. Começa com um modelo de escalonamento ideal que representa o comportamento ideal da política, para logo depois adicionar três componentes que representam a implementação real: $Overhead_j$(sobrecarga associado à mensagem), $Overhead_{sys}$(sobrecarga associada ao sistema), e $Blocking_i$(bloqueio por inversão de prioridades). Ao somar estes custos de escalonamento obtém-se um modelo genérico

de escalonamento do recurso.

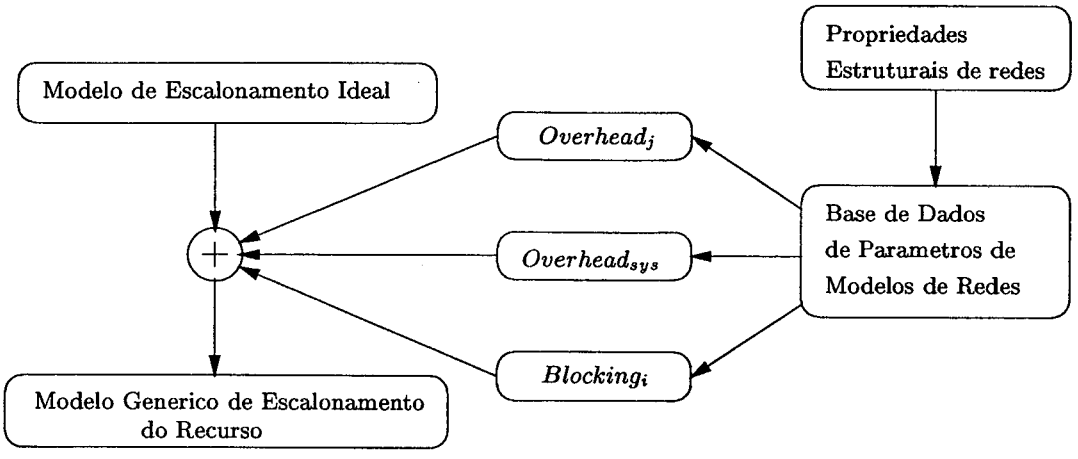

Figura 3.3: Modelo Genérico do Escalonamento do Recurso

O bloqueio por inversão de prioridade se refere ao efeito de preempção limitada de uma mensagem de prioridade maior que é atrasada na sua execução por uma mensagem de menor prioridade, no caso dependendo do mecanismo usado deve se limitar ao maior atraso no tempo de inspeção.

A contribuição da sobrecarga associado à mensagem se refere por exemplo ao empacotamento ou fragmentação das mensagens na execução dos procedimentos de transmissão e recepção das mensagens.

Por último, a contribuição da sobrecarga associado ao sistema se refere à troca de mensagens de controle, como por exemplo sincronismo de relógios, circulação de fichas, etc.

Considerando então os efeitos não ideais, a condição necessária e suficiente para que um conjunto de mensagens i de um nó k de uma rede com N nós, seja escalonável, dentro de um modelo de escalonamento genérico seria:

$$\begin{gathered} \forall i \quad \forall k \quad 1 \leq k \leq N \quad 1 \leq i \leq n \quad D_{i,k} \leq T_{i,k} \\ min_{0 \leq t \leq D_{i,k}} \sum_{j=1}^{i} \left\{ \frac{C_{j,k}+Overhead_{j,k}}{t} \left\lceil \frac{t}{T_{j,k}} \right\rceil \right\} + \frac{Overhead_{sys}}{t} + \frac{Blocking_{i,k}}{t} \leq 1 \end{gathered} \tag{3.8}$$

3.4.3 Abordagem para a análise de escalonabilidade baseada em EDF [Spu96]

Esta proposta baseia-se em sistemas escalonados pelo seu deadline (EDF) e propõe um teste para abordagens de escalonamento baseadas em EDF.

Similar à abordagem do Tindell, o objetivo é analisar sistemas, com tarefas periódicas, as quais podem ter deadlines arbitrários, release jitter, e podem compartilhar recursos. A análise se baseia no conceito de "busy period", que é o intervalo de tempo que o processador se mantém ocupado com tarefas de prioridade maior ou igual ao nível analisado.

Primeiro apresenta a condição de "feasibility"e depois o procedimento de cálculo do pior caso de tempo de resposta de uma tarefa com uma política de escalonamento EDF.

3.4.3.1 Modelo de computação e Notação

A proposta está inspirado nos resultados obtidos nos trabalhos do Tindell, substituindo o escalonamento preemptivo com prioridades fixas por um escalonamento preemptivo com uma política EDF.

Considera o escalonamento de tarefas em um único processador. As tarefas consistem de um infinito número de requisições, ou instâncias, cujos tempos de chegadas estão separadas por um mínimo tempo T, chamado de período. As instâncias das tarefas podem chegar em qualquer momento, mas a chegada tem que ser reconhecida por um despachante, que coloca a instância numa fila de pronto, onde se diz que é liberada. O tempo que transcorre da chegada da tarefa até sua liberação é chamado de "release jitter".

Cada instância da tarefa é caracterizada por um tempo limitado de computação C que deve ser completado antes do tempo D"deadline relativo"depois de sua chegada. A fila de pronto é ordenada em função do deadline de cada tarefa, por um despachante EDF("earliest deadline first") preemptivo. As tarefas podem compartilhar recursos através do protocolo "Priority Celing"ou "Stack Resource Policy".

A notação utilizada é a seguinte:

- C_i tempo de computação máximo da tarefa i em cada liberação
- D_i deadline da tarefa i, medição relativa ao tempo de chegada da tarefa
- B_i máximo bloqueio que a tarefa i pode experimentar devido à operação de um protocolo de controle de concorrência.
- J_i máximo *release jitter* da tarefa i
- r_i pior caso de tempo de resposta da tarefa i
- I_i número de instâncias da tarefa i liberadas antes do tempo t
- H_i número de instâncias da tarefa i com deadline menor ou igual ao tempo t

- $s_i(a)$ tempo de liberação da primeira instância da tarefa i, em um padrão de chegada onde outra instância i chega no tempo a
- L tamanho do primeiro "busy period"no padrão de chegada mais crítico

3.4.3.2 Teste básico de viabilidade(*feasibility*)

Em [Spu96] é apresentado um modelo simplificado que considera todas as tarefas com "release jitter"nulo, sem compartilhar recursos. Nesse modelo é desenvolvido um teste básico de viabilidade, para logo ser estendido para um modelo mais geral.

Dado um conjunto de tarefas, o pior padrão de chegada seria aquele no qual as instâncias de tarefas são liberadas o mais rápido possível,(*as soon as possible(asap)*), isto é, a primeira instância de todas as tarefas são liberadas no tempo *t=0*, e as outras instancias são liberadas de acordo com os períodos de suas tarefas.

No texto indicado é definido também um "busy period"com o pior padrão de chegada(*asap*). Esse "busy period"corresponde ao tempo de ocupação contínua do processador de prioridades maiores ou igual até o deadline considerado.

O tamanho L do "busy period"pode ser calculado através da fórmula iterativa:

$$\begin{cases} L^{(0)} & = \sum_{i=1}^{n} C_i, \\ L^{(m+1)} & = W(L^{(m)}), \end{cases} \tag{3.9}$$

onde $W(t)$ é a carga de trabalho cumulativo durante o tempo t:

$$W(t) = \sum_{i=1}^{n} \left\lceil \frac{t}{T_i} \right\rceil C_i$$

A computação da equação 3.9 se detém quando dois valores consecutivos são iguais, isto é, $L^{(m+1)} = L^{(m)}$, sendo L igual ao valor $L^{(m)}$. Para que a seqüência $L^{(m)}$ convergir a L em um número finito de passos, a utilização do sistema deve ser menor que a unidade, se esta condição não se cumpre, o conjunto de tarefas não é possível:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Considerando na carga acima aquelas tarefas que devem se executar antes ou em t, teremos um trabalho cumulativo dado por:

$$\sum_{D_i \leq t} \left(1 + \left\lfloor \frac{t - D_i}{T_i} \right\rfloor\right) C_i$$

A condição necessária e suficiente para a viabilidade de um conjunto de tarefas, é que todos os deadlines absolutos d no primeiro "busy period"superem os trabalhos cumulativos das tarefas que devem se completar nesse período:

$$d \geq \sum_{D_i \leq d} \left(1 + \left\lfloor \frac{d - D_i}{T_i} \right\rfloor\right) C_i \tag{3.10}$$

3.4.3.3 Encontrando o pior tempo de resposta

O pior caso de tempo de resposta r_i da instância de uma tarefa i é o máximo intervalo entre o tempo de chegada e o tempo em que se completa a instância da tarefa. A computação de r_i é importante quando analisamos um sistema distribuído porque o tempo de resposta de uma tarefa emissora é o máximo jitter experimentado pela mensagem enviada.

Nem sempre é encontrado o pior caso de tempo de resposta no primeiro "busy period", a idéia é que o tempo para completar uma instância de uma tarefa com deadline d, deve ser o final do "busy period"correspondente, no qual todas as instâncias executadas possuem deadlines menores ou iguais a d. Para encontrar o maior de tais períodos aplica-se o seguinte lemma [Spu96]:

Lema: o pior caso de tempo de resposta de uma tarefa i é encontrado no "busy period"no qual todas as outras tarefas são liberadas com o pior padrão de chegada(asap), ou seja, todas as tarefas tem suas instâncias liberadas simultaneamente no instante zero e apresentando suas máximas freqüências

O lemma anterior sugere o seguinte algoritmo de computação do pior tempo de resposta de uma tarefa i: é necessário calcular o tamanho do "busy period"de instâncias de tarefas com deadlines menores ou iguais que da instância considerada, com um padrão de chegada como se mostra na figura 3.4, ou seja, todas as tarefas menos i, são liberadas de maneira sincronizadas no tempo $t = 0$, na sua freqüência máxima. Nossa atenção é centrada na instância liberada no tempo $t = a$, onde $a \geq 0$, com possíveis instâncias anteriores, que contribuem a incrementar o tamanho do "busy period".

Para o caso em particular da figura, onde $a \geq 0$, o tempo de chegada da primeira instância da tarefa i:

$$s_i(a) = a - \left\lfloor \frac{a}{T_i} \right\rfloor T_i$$

Se todas as outras tarefas são liberadas inicialmente em $t = 0$, no tempo t, $\left\lceil \frac{t}{T_j} \right\rceil$ instâncias de j são liberadas, para cada $j \neq i$. Mas, no máximo $1 + \left\lfloor \frac{a+D_i-D_j}{T_j} \right\rfloor$ delas devem se completar antes de $a + D_i$. Então a carga de trabalho de maior prioridade ou igual a $d = a + D_i$ que chega até o tempo t é:

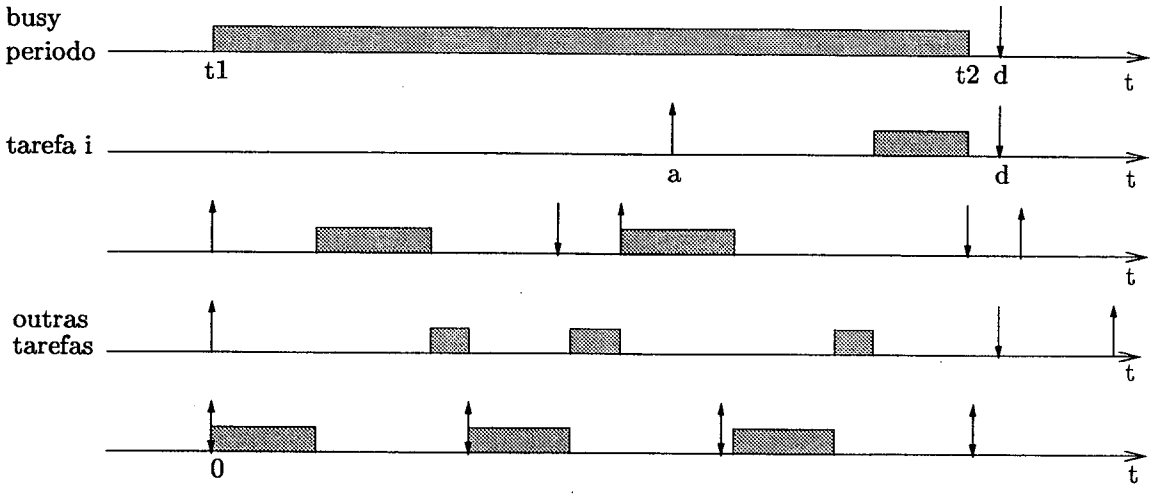

Figura 3.4: Busy período e padrão de chegada Asap

$$W_i(a,t) = \sum_{\substack{j \neq i \\ D_j \leq a + D_i}} min\left\{\left\lceil \frac{t}{T_j} \right\rceil, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor\right\} C_j + \delta_i(a,t) C_i,$$

onde

$$\delta_i(a,t) = \begin{cases} min\left\{ \left\lceil \frac{t - s_i(a)}{T_i} \right\rceil, \ 1 + \left\lfloor \frac{a}{T_i} \right\rfloor \right\} & se \ t > s_i(a), \\ 0 & outros \end{cases}$$

O tamanho $L_i(a)$ do "busy period"relativo ao deadline d pode ser computado com a seguinte fórmula iterativa:

$$\begin{cases} L_i^{(0)}(a) = \sum_{\substack{j \neq i \\ D_j \leq a + D_i}} C_j + I_{\{s_i(a)=0\}} C_i, \\ L_i^{(m+1)}(a) = W_i(a, L_i^{(m)}(a)), \end{cases} \tag{3.11}$$

onde

$$I_{\{s_i(a)=0\}} = \begin{cases} 1 & se \ s_i(a) = 0 \\ 0 & outros \end{cases}$$

A equação 3.11 converge em um finito número de passos se a utilização do sistema é menor que a unidade, ou seja:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

Com o valor de $L_i(a)$ determinado, pode-se encontrar o pior tempo de resposta relativo à a como:

$$r_i(a) = max\{C_i, L_i(a) - a\}$$

Finalmente, a pior tempo de resposta para uma tarefa i seria:

$$r_i = max_{a \geq 0}\{r_i(a)\} \tag{3.12}$$

Sendo L um limite superior, ou seja, é o máximo valor do "busy period", os valores de a para os quais temos que avaliar $r_i(a)$ estão dentro do intervalo $[0, L - C_i]$. Pode-se deduzir que o máximo $L_i(a)$ pode ser encontrado para aqueles valores de a tal que no padrão de chegada, existe ao menos uma instância de uma tarefa diferente de i com deadline igual a d, ou todas as tarefas estão sincronizadas, tendo uma instância de cada tarefa em $t = 0$, isto é, $s_i(a) = 0$.

Na figura 3.5 se mostra o pior tempo de resposta de uma tarefa i. Observe que na escala, para um tempo de chegada $a = 9$, o pior tempo de resposta não é alcançado no primeiro "busy period".

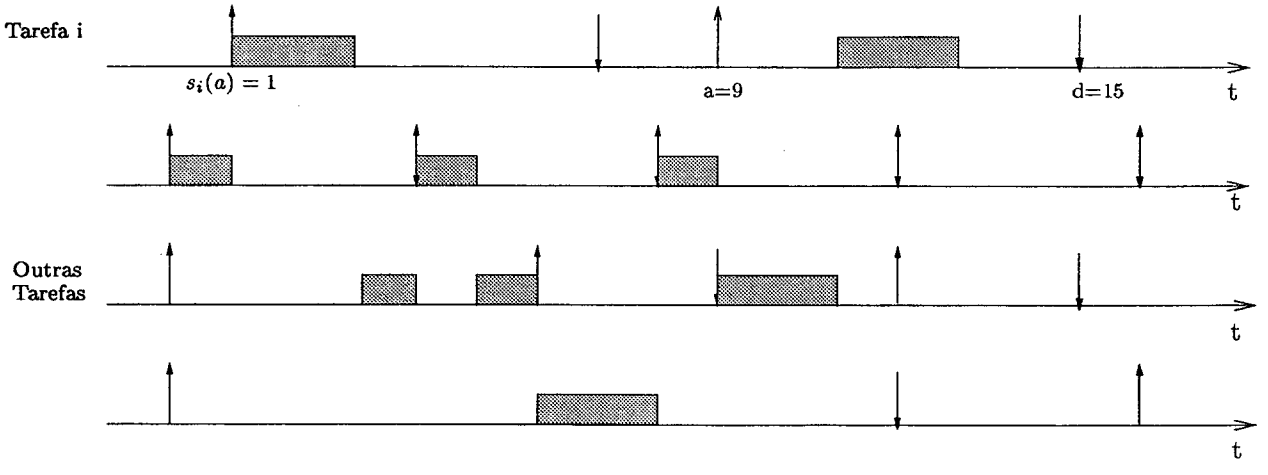

Figura 3.5: Pior tempo de resposta para a tarefa i com um tempo de chegada $a = 9$

3.4.3.4 Problema do release jitter

Se consideramos um modelo mais realista, onde o "realese jitter"não é nulo, assumindo que uma tarefa i depois de cada chegada pode ser demorada um tempo máximo J_i até ser liberada. A equação 3.9 deve ser modificada na definição de carga de trabalho acumulada $W(t)$:

$$W(t) = \sum_{i=1}^{n} \left\lceil \frac{t + J_i}{T_i} \right\rceil C_i$$

Para uma tarefa i, o número de instâncias com deadline menor ou igual a t, agora é:

$$1 + \left\lfloor \frac{(t + J_i) - D_i}{T_i} \right\rfloor$$

A equação 3.10 seria:

$$d \geq \sum_{D_i \leq t} \left(1 + \left\lfloor \frac{d + J_i - D_i}{T_i} \right\rfloor\right) C_i$$

O tempo de liberação da primeira instância:

$$s_i(a) = a + J_i - \left\lfloor \frac{a + J_i}{T_i} \right\rfloor T_i$$

A carga de trabalho de maior prioridade que se executa até o tempo t agora é dada por:

$$W_i(a,t) = \sum_{\substack{j \neq i \\ D_j \leq a + D_i + J_j}} min\left\{\left\lceil \frac{t+J_j}{T_j} \right\rceil, 1 + \left\lfloor \frac{a+D_i+J_j-D_j}{T_j} \right\rfloor\right\} C_j + \delta_i(a,t)C_i,$$

onde

$$\delta_i(a,t) = \begin{cases} min\left\{ \left\lceil \frac{t-s_i(a)+J_i}{T_i} \right\rceil, \; 1 + \left\lfloor \frac{a+J_i}{T_i} \right\rfloor \right\} & se \;\; t > s_i(a), \\ 0 & outros \end{cases}$$

O tamanho do resultante "busy period"relativo ao deadline $d = a + D_i$, pode então ser computado através da equação 3.11, onde somente a definição do valor inicial deve ser mudado:

$$L_i^{(0)}(a) = \sum_{\substack{j \neq i \\ D_j \leq a + D_i + Jj}} C_j + I_{\{s_i(a)=0\}} C_i,$$

O pior tempo de resposta relativo à a agora é:

$$r_i(a) = max\{J_i + C_i, L_i(a) - a\}$$

Existe umà diferença entre o significado da variável a quando não se considera o release jitter e quando é considerado. No primeiro caso o tempo de chegada coincide com o tempo de liberação (a). No segundo caso o tempo de chegada e igual à a e o tempo de liberação seria J_i unidades mais tarde. Os valores significativos de a, utilizados no calculo de $r_i(a)$ são os que estão no intervalo $[-J_i, L - J_i - C_i)$ Finalmente, o pior tempo de resposta para uma tarefa i seria:

$$r_i = max_{a \in [-J_i, L-J_i-C_i)}\{r_i(a)\}$$

3.4.3.5 Compartilhando recurso

Outro termo adicional a ter presente na análise é o bloqueio por inversão de prioridade. O bloqueio deve ser limitado a uma duração máxima por "busy period", limitado por B_i.

O termo L do primeiro "busy period" não é afetado na presença de bloqueio. O bloqueio somente desvia a escala, não afeta a carga de trabalho que chega até o tempo t. A carga de trabalho somente depende do padrão de liberação. Então L é calculado pela equação 3.9. Para verificar os deadline da equação 3.10 temos que considerar o fator de bloqueio:

$$d \geq \sum_{D_i \leq t} \left(1 + \left\lfloor \frac{d + J_i - D_i}{T_i} \right\rfloor\right) C_i + B_{i(d)} \tag{3.13}$$

Em [Spu96] são discutidos os detalhes da equação 3.13.

Na equação 3.11 temos que considerar o possível tempo adicional de bloqueio na computação de L_i^{m+1}

$$L_i^{(m+1)}(a) = W_i(a, L_i^{(m)}(a)) + B_{i(a+D_i)} \tag{3.14}$$

O pior tempo de resposta relativo à a agora é:

$$r_i(a) = max\{J_i + C_i + B_i, L_i(a) - a\}$$

3.5 Seleção da Abordagem para a Análise de Escalonabilidade

A análise de escalonabilidade utilizando diferentes algoritmos de atribuição das prioridades tem sido motivo de muitos trabalhos e propostas. Começando com propostas para hipóteses simplificadas, que foram incorporando situações mais realistas e complexas, aumentando de forma proporcional a sofisticação e complexidade dos testes de escalonabilidade. Mas isto é em resposta a uma necessidade das aplicações, obrigando a incrementar o espectro das situações que se querem representar. Por isto observa-se uma tendência nos últimos anos na aparição de ferramentas que auxiliem neste sentido, criando um suporte computacional para a automatização dos testes de escalonabilidade, exemplos delas são o "STRESS", simulador de escalonamento desenvolvido pela Universidad de York, o

"SEW" desenvolvido pela Carnegie Mellon University, com teste de escalonabilidade para sistemas multimídia e a proposta no trabalho [ABNS98] que trata-se de uma ferramenta para aplicações críticas.

A seleção de um teste ou outro para ser utilizado na análise de escalonabilidade de um conjunto de tarefas, resulta complicado. As abordagens que utilizam uma política de prioridades fixas foram historicamente mais tratadas na literatura, começando a cobrar importância nos últimos tempos as abordagens com uma política de atribuição de prioridades dinâmicas.

As três propostas para a análise apresentadas neste trabalho, se caracterizam por possuir testes de escalonabilidade exatos. Sendo necessária a correta identificação das propriedades e características temporais do modelo de escalonamento tais como interferências, sobrecargas associadas às mensagens(produto do empacotamento e fragmentação), sobrecargas associadas ao sistema como conseqüência da execução do protocolo e o bloqueio por inversão de prioridade. Nestas propostas, a garantia de entrega das mensagens se obtém a partir de uma carga limitada, do conhecimento prévio das mensagens, do modelo de recursos e da política de tempo real. Como não é possível saber em tempo de projeto, exatamente qual mensagem vai ser transmitida, ou seja, que não se tem um conhecimento antecipado do comportamento temporal, o que se garante é o atendimento dos requisitos temporais das mensagens expressados na forma de deadlines.

Uma vez apresentadas as similitudes passemos a discutir as diferenças das propostas apresentadas. A técnica baseada em interferências apresentada em [TBW94] e a baseada em EDF apresentada em [Spu96] admitem um modelo de mensagens mais flexível no sentido de permitir deadlines menores, maiores ou iguais aos receptivos períodos. Entretanto, a técnica baseada em saturação proposta em [KS94] e [KSS95b] restringe o modelo a mensagens com deadlines menores ou iguais a seus períodos.

Outra diferença se refere à utilização do sistema. As técnicas que utilizam uma política de prioridades fixas, no caso seriam a baseada em interferências e em saturação, apresentam uma menor utilização do sistema que a técnica que utiliza uma política de atribuição de prioridades dinâmicas.

No trabalho [FFF95] e [Fon97] se apresenta um comparativo das abordagens baseadas em interferências e em saturação, e uma vantagem desta última é a forma de sua construção, sendo mais clara na representação, facilitando a representação das propriedades do protocolo no teste, a influência das sobrecargas sobre a escalonabilidade do conjunto de mensagens, sendo mais fácil também, detectar os casos o situações críticas de escalonabilidade. Tanto em [TBW94], como em [Spu96] são consideradas ditas sobrecargas, somente muda a forma de representação.

Para definir o teste de escalonabilidade que se utilizaria no presente trabalho, existia uma inclinação por o EDF como política de atribuição de prioridades, devido por um lado, a que alcança uma maior utilização do recurso, se comparamoss com o escalonamento de prioridades fixas, e por otro lado, que as prioridades dinâmicas tem sido menos tratadas na literatura.

Definida a política de atribuição de prioridades, optou-se por a abordagem apresentada em [Spu96] como técnica adotada para a análise de escalonabilidade já que, como a proposta do [TBW94], se apresenta como uma proposta flexível, no sentido do modelo de mensagens que admitem, além de que o EDF alcança uma maior utilização do recurso se comparamos com o escalonamento de prioridades fixas.

3.6 Custo de quantização produto de um limitado número de níveis de prioridade

Existe um custo a considerar no escalonamento de tarefas tempo real, refere-se ao custo como conseqüência do limitado número de níveis de prioridades existentes no escalonador.

No trabalho [MNS96] é proposto uma metodologia que mapeia deadlines em valores de prioridades, procurando minimizar o efeito de inversão de prioridade devido a esse mapeamento.

Primeiramente nesse trabalho é apresentado o mapeamento usual de deadline. O eixo de tempo é dividido em unidades de tempo constante, a granularidade que define um máximo erro U na maneira como é expressa o deadline.

3.6.1 Custo de quantificação das restrições temporais para uma escala de tempo com unidades de tempo constante

- **Cálculo da prioridade**

  A forma natural de computar a prioridade das mensagens é:

$$p_i = \left\lfloor \frac{d_i - T_{start}}{U} \right\rfloor$$

  Pode-se deduzir da expressão acima que:

  - a prioridade depende de d_i - T_{start}, onde d_i é o deadline absoluto da mensagem m_i, e T_{start} é o tempo de começo da disputa pelo canal.

- no pior caso a prioridade sofre um erro de quantização U.
- como conseqüência da granularidade, um mesmo valor de prioridade pode ser atribuído a várias mensagens (estão localizadas no mesmo intervalo de tempo), provocando possíveis inversões de prioridades.

- **Cálculo da unidade de tempo**

  Com n bits de prioridade, os níveis de prioridade podem ser no máximo 2^n, e com o maior deadline relativo à T_{start}, D_{max}, a escolha de U pode ser:

$$U = \frac{D_{max}}{2^n}$$

  definindo desta forma a granularidade de tempo igual à constante acima.

- **Custo de quantização**

  Para medir essa quantificação do eixo de tempo, é definido um novo parâmetro, denominado custo de quantização:

$$x_i = \frac{U}{D_i}$$

  O parâmetro x_i representa uma penalidade quando a computação do deadline se faz com um erro igual U. Significa que se $x_i > 1$ o custo é de uma dimensão tal que não garante a escalonabilidade da mensagem analisada. Expressado de outra forma, quer dizer que o tamanho de dita granularidade se comparar-mos com o deadline analisado, é tanto maior que se estaria produzindo o efeito não desejado de inversão de prioridades, mas em uma magnitude tal que o sistema não garante a restrição temporal da mensagem.

  Se substituímos U na expressão de x_i, das equações acima temos:

$$x_i = \frac{D_{max}}{2^n D_i}$$

  e como k_i^{max} é:

$$k_i^{max} = \frac{D_{max}}{D_i}$$

  A relação entre n e k_i^{max} então é:

$$x_i = \frac{k_i^{max}}{2^n}$$

De aqui temos que a relação k_i^{max}, que da uma idéia da carga do sistema, não deve superar os 2^n níveis de prioridades disponíveis no sistema, caso contrário não se garante a escalonabilidade do conjunto de mensagens. Enquanto o número de níveis de prioridades se mantenha baixo ou/e a relação de D_{max} e D_{min} aumente, se faz mais difícil garantir a escalonabilidade dessa carga.

3.6.2 Solução com inversão de prioridade limitada

Nesta seção apresenta-se a técnica para minimizar o custo de quantização, para isso o eixo do tempo é dividido em seções que se incrementam de forma exponencial.

- **Cálculo da unidade de tempo**

  Um máximo custo q é fixado a priori, tal que $1/q$ deve ser inteiro. Dividindo cada seção em $1/q$ unidades de tempo

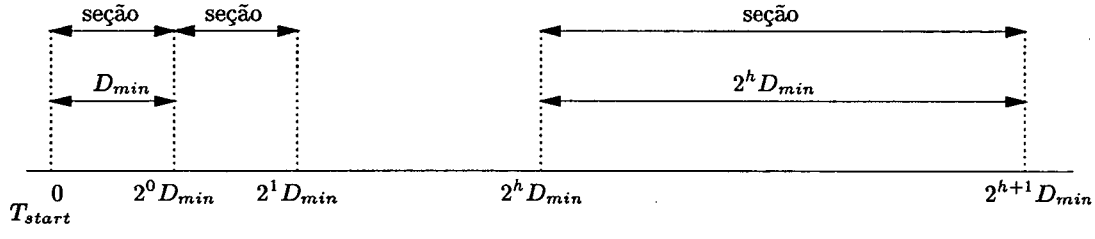

Figura 3.6: Distribuição das unidades de tempo no eixo do tempo

Considerando a primeira seção de tamanho igual a D_{min}:

- a unidade de tempo da primeira seção é igual a:

$$U = \frac{D_{min}}{1/q}$$

A segunda seção $[D_{min}, 2D_{min}]$ se divide em 1/q unidades de tempo, então:

- a unidade de tempo da segunda seção é igual a:

$$U_0 = 2^0 \frac{D_{min}}{1/q} = \frac{D_{min}}{1/q} = U$$

De forma genérica cada seção $[2^h D_{min}, 2^{h+1} D_{min}]$ é dividida em 1/q unidades de tempo:

– a unidade de tempo é igual a:

$$U_h = \frac{2^{h+1}D_{min} - 2^h D_{min}}{1/q} = \frac{2^h D_{min}}{1/q} = 2^h U$$

Pode-se observar que a unidade de tempo depende do lugar onde se encontra o deadline relativo ($d_i - T_{start}$). Se o deadline relativo aumenta, a unidade de tempo aumenta, aumentando por tanto a granularidade.

- **Cálculo da prioridade**

Para calcular a prioridade correspondente ao intervalo [d_i - T_{start}], é preciso encontrar o lugar no eixo de tempo. A computação se faz através de dois passos, primeiro se avalia h:

$$h = \begin{cases} \lfloor \log_2 \frac{d_i - T_{start}}{D_{min}} \rfloor & se \quad d_i - T_{start} \geq D_{min} \\ 0 & outros \end{cases}$$

Com o valor de h a prioridade se calcula como:

$$p = h1/q + \left\lfloor \frac{d_i - T_{start}}{2^h U} \right\rfloor \tag{3.15}$$

onde

$$U = \frac{D_{min}}{1/q}$$

- **Custo de quantização**

O deadline de uma mensagem m_i se encontra no intervalo genérico [$2^{h_i}D_{min} \leq D_i < 2^{h_i+1}D_{min}$]. O máximo erro de quantização vai-se corresponder com a unidade de tempo do intervalo onde D_i se localiza. A unidade de tempo é $2^{h_i}U$.

Para cálcular o custo para uma escala exponencial, quando o calculo da prioridade se faz com o máximo erro, isto é, $2^{h_i}U$, temos que:

$$x_i = \frac{2^{h_i}U}{D_i}$$

Encontrando um limite superior deste custo temos:

$$x_i = \frac{2^{h_i}U}{D_i} \leq \frac{2^{h_i}U}{2^{h_i}D_{min}} = \frac{U}{D_{min}} = q$$

então

$$sup_i(x_i) = q$$

A continuação define-se k_{max} como a máxima carga suportada pelo sistema, para logo depois encontrar a relação entre x, k_{max} e n:

$$k_{max} = max_i(k_i^{max}) = \frac{D_{max}}{D_{min}}$$

Em [MNS96] se detalham as transformações algébricas pelas quais se obtém a expressão:

$$x = \frac{1}{\left\lfloor \frac{2^n}{\lfloor \log_2 k_{max} \rfloor + \frac{k_{max}}{2^{\lfloor \log_2 k_{max} \rfloor}}} \right\rfloor} \tag{3.16}$$

Podemos concluir que com a divisão do eixo de tempo em intervalos que se incrementam exponencialmente, sendo que cada intervalo apresenta uma granularidade diferente, e um erro de quantização diferente, que aumenta exponencialmente, se consegue diminuir o peso do termo k_{max} dentro da expressão de calculo do custo de quantização, mantendo o mesmo número de níveis de prioridades proposto para o caso de uma escala de tempo com unidades de tempo constante.

Em [MNS96] se mostra um exemplo de um sistema com uma carga que não consegue garantir sua escalonabilidade utilizando uma escala de tempo com unidades de tempo constante, no entanto, utilizando a divisão do eixo em intervalos que se incrementam exponencialmente, se consegue garantir a escalonabilidade da mesma carga, para o mesmo número de níveis de prioridades.

3.7 Conclusões

Neste capítulo foram apresentadas algumas das diferentes técnicas que poderiam ser utilizadas para a análise de escalonabilidade e a justificativa da escolha de uma delas para o problema aqui proposto.

Outra questão importante considerada neste capítulo é a metodologia proposta em [MNS96] para minimizar o efeito de inversão de prioridade produto do limitado número de níveis de prioridades existente.

Com a apresentação dos aspectos gerais referidos à comunicação em sistemas tempo real e escalonamento de mensagens em ditos sistemas, estamos em condições de especificar

a camada de enlace para um sistema de comunicação tempo real, sendo esta temática abordada nos próximos capítulos, começando pela camada MAC e finalizando pela camada LLC.

Capítulo 4

Camada MAC utilizando o protocolo determinista CSMA-DCR

4.1 Introdução

Dentro dos protocolos apresentados no capítulo 2 que fornecem garantia de entrega das mensagens, foi selecionado como protocolo para este sistema de comunicação o CSMA-DCR(Carrier Sense Access Method/Deterministic Collision Resolution), que pode ser classificado como um protocolo que utiliza a abordagem de circuito virtual tempo real, ou seja, garante um tempo de serviço limitado, sendo que este tempo está composto por dois elementos, o tempo de acesso ao canal físico e o tempo de transmissão do pacote.

Este protocolo proposto em [LR93] é totalmente compatível com o protocolo Ethernet standard (ISO/OSI 8802/3-Ethernet standards) ou seja, interfaces convencionais que utilizam o protocolo CSMA/CD e interfaces com protocolos CSMA-DCR podem coexistir e trocar mensagens compartilhando o mesmo meio de comunicação. A diferença consiste em que o algoritmo de espera aleatória exponencial truncada é substituído por um algoritmo determinístico de busca em árvore binária balanceada. Esta é a principal característica do protocolo e é ela a que permite sua utilização para sistemas tempo-real. Outra característica importante é que utiliza largura de banda somente após a colisão, a diferença de outros protocolos que evitam a colisão dividindo a largura de banda.

Em quanto não se produz colisão o protocolo funciona como o CSMA/CD. Em caso de colisão, o CSMA-DCR dicotomiza de forma iterativa ao conjunto de fontes de mensagens, sendo necessário o conhecimento em tempo de configuração, do limite superior de fontes de mensagens. O período de resolução de uma colisão se denomina "época".

Existem diferentes opções do protocolo CSMA-DCR [LR93], que são selecionadas no

tempo de configuração.

- Um primeiro que indica o modo de busca na árvore, ou seja, se é "general" começa a busca pela raiz, e se é "leaf" começa pelas folhas (neste caso pode começar pelos índices maiores ou pelos índices menores).
- Um segundo que indica o modo selecionado para entrar na "época", ou seja, se é "bloqueado" somente as mensagens envolvidas na colisão original poderão transmitir na "época". Se é "aberto", participarão da "época" as mensagens envolvidas na colisão original, mais aquelas que chegam mais tarde, mas em tempo, isto é, antes que seus índices sejam alcançados.

A interface utilizada para implementar o protocolo MAC neste trabalho está configurada no modo "leaf"e bloqueado.

4.2 Descrição do Protocolo de busca em árvore binária balanceada [LR93]

A cada estação ou fonte de mensagens é atribuído um índice, e como o protocolo baseia-se neste índice para dar um ordem na resolução da colisão, ou seja, os nós com menor índice transmitem primeiro, neste trabalho associamos dito índice com a prioridade. As prioridades são atribuídas a cada estação e não às mensagens.

Como requisito cada estação deve conhecer:

- O estado do barramento que pode ser:
  - ocupado com transmissão normal
  - ocupado com colisão
  - livre
- Seu próprio índice.
- Número total de índices consecutivos alocados às estações(Q).

Define-se q como o tamanho da árvore binária, que é a menor potência de 2 maior ou igual a Q. Até a ocorrência de uma colisão o protocolo opera como o CSMA/CD. Após a colisão, é que começa o período de resolução da colisão ou "época".

Todas as estações conhecem seu índice, e são capazes de "escutar" o canal para identificar o estado dele. Após a colisão as estações são divididas em dois grupos, "winners"(com

a metade das estações com índice mais baixo) e "losers"(com a outra metade de estações com os índices mais altos). As "winners" tentam transmitir novamente. Se ocorre uma nova colisão, este grupo se divide novamente, isto até que fique uma estação única no grupo "winners", que transmite seu quadro de dados. As estações do grupo "losers" aguardam termino de transmissão bem sucedida de outro nó, seguida de meio livre. No caso que o grupo "winners" estiver vazio, a busca é revertida, isto é, se realiza uma nova subdivisão a partir do último grupo "losers". A "época" é encerrada quando todas as estações envolvidas na colisão inicial conseguem transmitir seus dados.

Consideremos o seguinte exemplo, uma rede com 16 estações, das quais 6 tentam transmitir simultaneamente, produzindo uma colisão. Na figura 4.1 se observa a árvore binaria balanceada para este caso. Os números entre colchetes indicam os índices das estações que formam dito grupo, o número dentro do círculo indica a ordem cronológica. Depois da primeira colisão das estações 2,3,5,12,14 e 15 as estações são divididas em dois grupos, um com as estações de 0 até 7("winners") e outro com as estações de 8 até 15("losers"). As "winners"tentam transmitir, mas acontece outra colisão das estações 2,3,e 5, se divide novamente, e assim vai se repetindo o processo até que todas as estações envolvidas na colisão inicial conseguem transmitir.

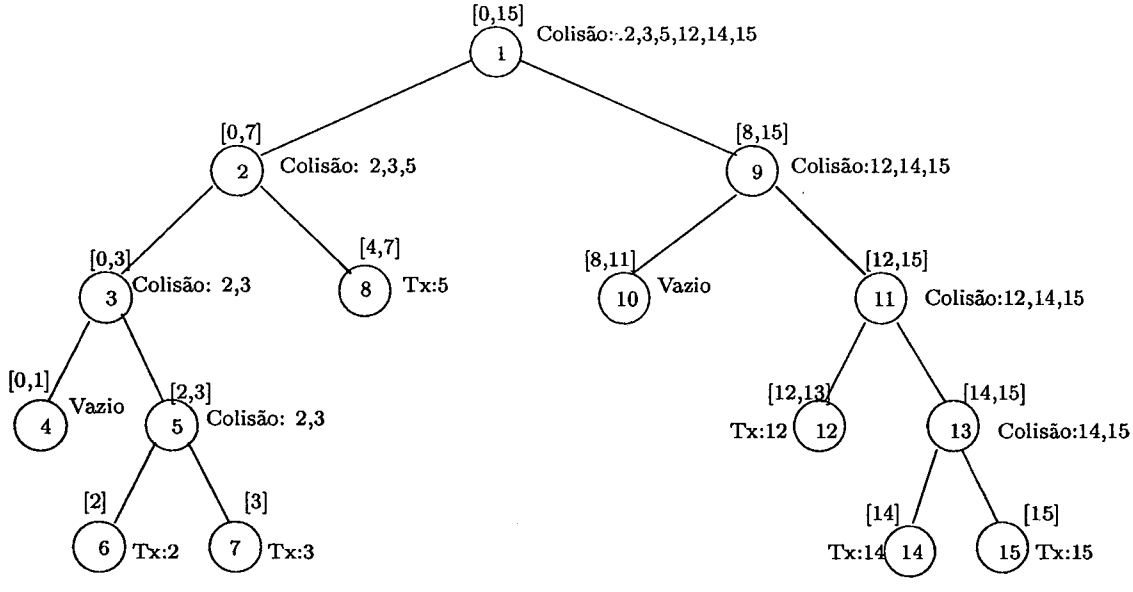

Figura 4.1: Árvore binária balanceada para Q = 16

Esta tabela é outra forma de representar o mesmo exemplo anterior, mas que apresenta de uma forma mais clara a cronologia de como se sucede dita "época". O tempo que transcorre durante um canal vazio ou canal ocupado é de um slot-time, considerado neste exemplo como 40 microsegundos e o tempo que transcorre para uma transmissão de um

quadro é igual a 6 slot-times.

ordem cronológica	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
estado do canal	C	C	C	V	C	X	X	X	C	V	C	X	C	X	X
índice de estação	2	2	2		2	2	3	5	12		12	12	14	14	15
	3	3	3		3				14		14		15		
	5	5							15		15				
	12														
	14														
	15														

C = canal ocupado com colisão
V = canal vazio ou livre
X = canal ocupado com transmissão normal

Continuando com o mesmo exemplo, se se quer calcular o tempo que a estação com índice 12 deve esperar para dar inicio a sua transmissão, se verifica na tabela a quantidade de colisões, vazios e transmissões que acontecem até que a estação 12 adquire o direito a transmitir, esto es:

$$T_{espera}(12) = 6\ colisões{+}2\ vazios{+}3\ transmisões = (6)(40){+}(2)(40){+}(3)(6)(40) = 680\ \mu seg$$

Se se quer calcular a duração desta "época" verifica-se na tabela o total de colisões, de vazios e de transmissões:

$$T_{época} = 7\ colisões{+}2\ vazios{+}6\ transmisões = (7)(40){+}(2)(40){+}(6)(6)(40) = 1800\ \mu seg$$

O caráter determinista do protocolo vem de fato de se poder calcular a duração de uma "época".

4.3 Análise do pior caso na latência de mensagens

Para calcular o pior caso na latência de mensagens, deve-se considerar o pior cenário, ou seja quando todas as estações na rede tentam transmitir ao mesmo tempo, para calcular este tempo define-se os seguintes parâmetros:

- Q = número total de índices consecutivos alocados às estações.

- $\varphi(k)$ = número de ramos da árvore binária percorridos para uma mensagem proveniente de um nó com índice k.
- q = menor potência de 2 maior ou igual ao maior índice disponível
- $\sigma(k)$ = número de potências de 2 contidas em k.
- s = 1 slot-time(2 vezes o tempo de propagação do sinal na rede).
- ρ = tempo máximo de transmissão de uma mensagem no meio físico (depende do comprimento da mensagem em bits e da taxa de transmissão).

O protocolo CSMA-DCR de forma geral, como se mostra na figura 4.2 apresenta dois tempos, o tempo de busca na árvore binária do próximo nó que obtem o direito a transmitir no canal compartilhado e o tempo de transmissão dos pacotes envolvidos na colisão.

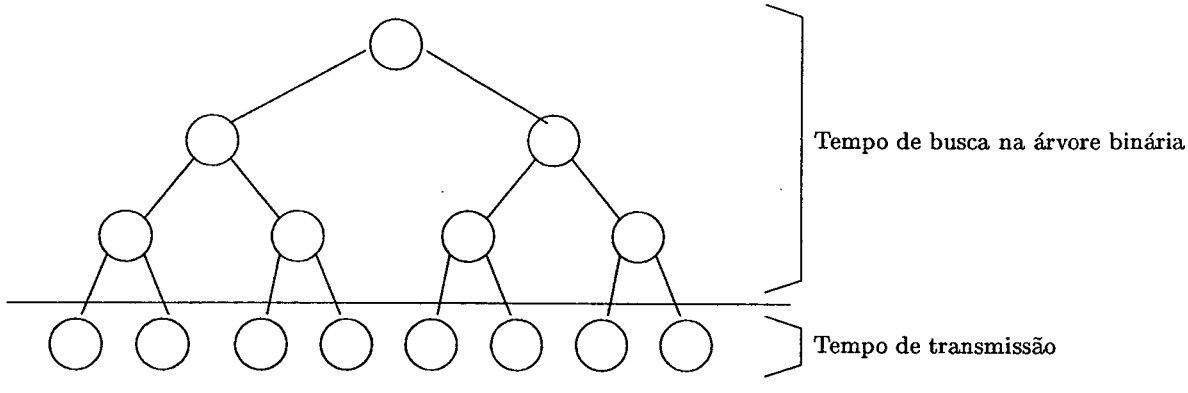

Figura 4.2: Tempos da árvore binária

A continuação definiremos alguns tempos característicos do protocolo MAC CSMA-DCR. O tempo de busca na árvore binária para uma mensagem proveniente de um nó k é:

$$T_{busca}(k) = \varphi(k)s$$

É importante destacar a diferença entre o tempo de busca e o tempo que uma estação deve esperar para iniciar sua transmissão, este último inclui o tempo de busca na árvore binária mais o tempo de transmissão das estações com índice menor, isto pode ser expressado como:

$$T_{espera}(k) = \varphi(k)\ s\ +\ k\ \rho$$

Para uma mensagem participando de uma dada época, temos que:

$$\varphi(k) = \log_2 q + k - \sigma(k)$$

Para o exemplo anterior, tomando uma mensagem da estação 12, temos que:
- $q = 16 \quad k = 12$
- $\sigma(12) = 2$ porque $(12 = 2^2 + 2^3)$
- portanto $\varphi(12) = \log_2 16 + 12 - 2 = 14$

Assumindo $s = 40$ microsegundos e $\rho = 6s$, obteremos para o pior caso de tempo de espera da mensagem da fonte com índice 12 o valor de 3400 microsegundos.

$$T_{espera}(12) = 14s + 11\rho = 14(40) + (11)(6)(40) = 3400\mu seg$$

O tempo máximo de uma "época"é aquele em que todas as estações possuem uma mensagem de tamanho máximo para transmitir, composto pelo tempo máximo de busca na árvore binária e pelo tempo máximo de transmissão. Pode-se representar pela seguinte equação :

$$T_{época} = T_{busca\ máx} + T_{tx\ máx} = \varphi(q-1)\ s + Q\ \rho \tag{4.1}$$

Sendo que o primeiro termo representa o tempo gasto na busca dos ramos da árvore e o segundo termo é o tempo de transmissão das Q mensagens de tamanho máximo.

O adaptador de comunicação utiliza o modo *bloqueado* para entrar na "época", ou seja, participam dela somente as mensagens envolvidas na colisão original.

Então o máximo tempo na latência de uma mensagem seria no caso em que a mensagem chegue na fila de emissão no instante imediatamente após o inicio de uma época máxima, da qual ela ainda não faz parte. Neste caso, o pior caso de tempo de latência é:

$$T_{latência} = T_{época\ máx} + \varphi(k)\ s + k\ \rho \tag{4.2}$$

O tempo de latência depende do lugar que ocupa a estação na árvore binária, determinado pelo índice k. Obviamente as estações com índice maior terão um tempo de latência maior, e é por isto que o índice de certa forma representa uma prioridade global.

4.4 Conclusões

Neste capítulo se apresenta o protocolo CSMA-DCR para a camada MAC. A característica principal deste protocolo é sua natureza determinista, sendo este o motivo de sua escolha para sistemas com restrições temporais.

Podemos citar como características principais deste protocolo:

- a possibilidade de poder expressar o tempo máximo de latência de uma mensagem em função do índice alocado pela estação.
- pode-se calcular o tempo máximo de uma época numa situação de instante crítico.
- até que se produz uma colisão o protocolo funciona como um CSMA/CD clássico.
- ocupa largura de banda a partir que se produz uma colisão.

A placa que implementa o protocolo CSMA-DCR é compatível com placas de rede que utilizam o protocolo CSMA/CD, portanto no caso de não ativar o modo determinista, funciona como uma placa de rede com protocolo CSMA/CD clássica.

Uma vez especificada a camada MAC está-se em condições de definir a camada LLC para um serviço sem conexão e sem reconhecimento e realizar a análise de escalonabilidade para um protocolo CSMA-DCR. Isto será abordado no próximo capítulo.

Capítulo 5

Proposta de uma camada LLC para comunicação tempo real

5.1 Introdução

Neste capítulo se apresenta uma proposta de uma camada LLC para comunicação tempo real. Por motivos de simplicidade, optou-se por um serviço sem conexão e sem reconhecimento.

É importante destacar que a camada LLC que se especifica neste trabalho, embora seja uma camada que fornece um serviço sem conexão e sem reconhecimento, forma parte ao mesmo tempo, de um sistema de comunicação tempo real com garantia de entrega das mensagens. Para lograr isto, em tempo de projeto, um teste de escalonabilidade avalia se existe ou não possibilidade de alguma mensagem perder seu deadline e em tempo de execução, um escalonador preemptivo escolhe a mensagem com a prioridade mais alta para ser transmitida.

Para definir esta proposta de camada LLC, neste capítulo serão especificados aspectos tais como as primitivas de serviço, o protocolo de enlace para este tipo de serviço, e a análise de escalonabilidade das mensagens que se querem transmitir.

Dentro das primitivas que especificam o serviço sem conexão e sem reconhecimento, apresenta-se tanto as que formam parte da interface entre a camada de aplicação e a camada LLC, como as de a interface entre a camada LLC e a camada MAC. Se descrevem também, os procedimentos derivados deste tipo de serviço, para finalmente apresentar a análise de escalonabilidade das mensagens, para as quais se requer garantia de entrega. Para realizar a análise de escalonabilidade das mensagens se utiliza a técnica proposta por Spuri em [Spu96], para um protocolo MAC do tipo CSMA-DCR proposto por Le

Lann em [LR93].

A subcamada LLC de maneira geral é responsável por diferentes funções tais como: gerenciamento das conexões, controle de erro, controle de fluxo no enlace, fragmentação, realiza a função de multiplexação [SLC95] de acesso ao meio físico, através da definição de Pontos de Acesso a Serviços (Service Access Points - SAPs), e assim como também realizar o escalonamento das mensagens recebidas da camada de aplicação. Mas como o tipo de serviço utilizado nesta proposta é um serviço sem conexão e sem reconhecimento, não se realizam as funções de gerenciamento das conexões, controle de erro e controle de fluxo no enlace.

Para explicitar como é que se realiza a função de multiplexação mencionada anteriormente, observe na figura 5.1 como na camada de enlace existem dois níveis de endereçamento: o endereço MAC que identifica um ponto de conexão física à rede, e o SAP LLC que identifica um usuário do nível de enlace, permitindo assim a realização da multiplexação.

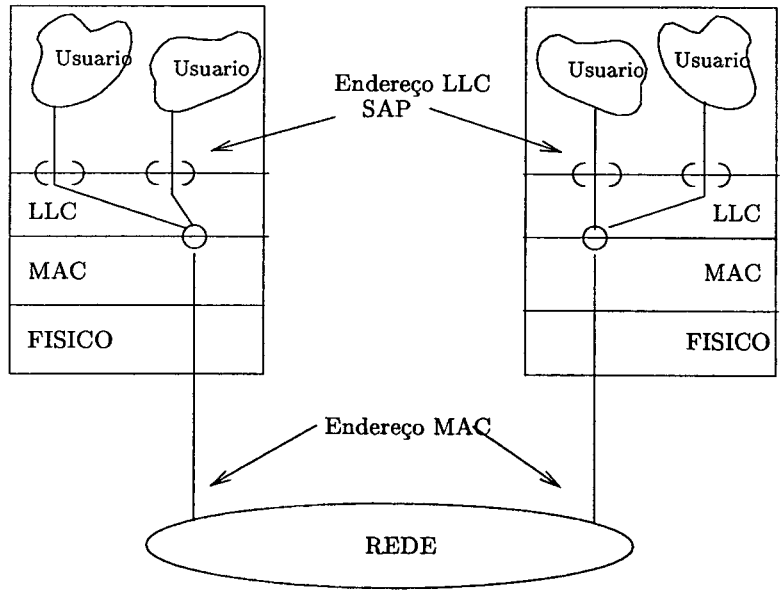

Figura 5.1: Endereços LLC e MAC

As outras funções da camada LLC são explicadas de forma indireta com a especificação da camada LLC abordada nesta proposta.

5.2 Projeto de uma camada LLC para um serviço sem conexão e sem reconhecimento

No serviço sem conexão e sem reconhecimento, as abordagens de escalonamento são no sentido melhor esforço. Como a nossa maior preocupação é a construção do teste de escalonabilidade modelando recursos concretos, para dominar a complexidade, a ideia é começar com serviços mais simples, com o objetivo de não inviabilizar as analises de escalonabilidade, assumindo que o serviço sem conexão e sem reconhecimento não envolve perdas de mensagens. Na prática isso poderia ser conseguido usando técnicas de replicação a nivel de MAC e físico.

O objetivo deste trabalho é o desenvolvimento de uma camada LLC, que embora utilize um serviço sem conexão e sem reconhecimento, forme parte de um sistema de comunicação com garantia de entrega das mensagens. A garantia no sentido do cumprimento das restrições temporais das mensagens. Esta garantia é obtida sobre a base de uma hipótese de carga, e de considerar a premissa que não existem perdas ou falhas das mensagens transmitidas. As condições normalmente aceitas para tal garantia envolvem uma carga estática, ou seja, limitada e conhecida em tempo de projeto e a reserva de recursos para a transmissão das mensagens no pior caso.

Para compensar a falta de mecanismos de controle de erro no serviço sem conexão e sem reconhecimento utilizado nesta proposta, se especifica uma primitiva cuja função é indicar o sucesso ou falha da requisição de serviço à camada MAC, retornando o estado à entidade LLC local. Outra alternativa é usar replicação da interface de rede utilizada, além do recurso próprio de controle de erro da interface que implementa o protocolo MAC, o FCS(Frame Check Sequence), que é um algoritmo de redundância cíclica(CRC - Cyclic Redundancy Check), cujo valor é calculado em função dos conteúdos dos campos que conformam o quadro MAC.

Na especificação desta camada de enlace se faz uso do padrão ANSI/IEEE 802.2(ISO 8802-2) [fSEC94], dentro do padrão IEEE 802.3 especificou-se o tipo de serviço sem conexão e sem reconhecimento, ou seja uma operação tipo 1. A vantagem de utilizar um padrão para a especificação da camada LLC é que a camada de aplicação independe da forma como foi desenvolvida a camada LLC, tendo somente que preocupar-se com a interface específica para o tipo de serviço solicitado.

Se utilizará a figura 5.2 como base para apresentar a camada LLC, mostrando como a entidade LLC basicamente estaria formada por um componente de transmissão, que realiza a fragmentação da mensagem(caso seja necessária) e o escalonamento das mensagens, um componente de recepção que realiza a blocagem dos pacotes recebidos para formar a

mensagem e as primitivas que especificam o tipo de serviço que está-se solicitando, tanto da interface com a camada de aplicação, como a interface com a camada MAC.

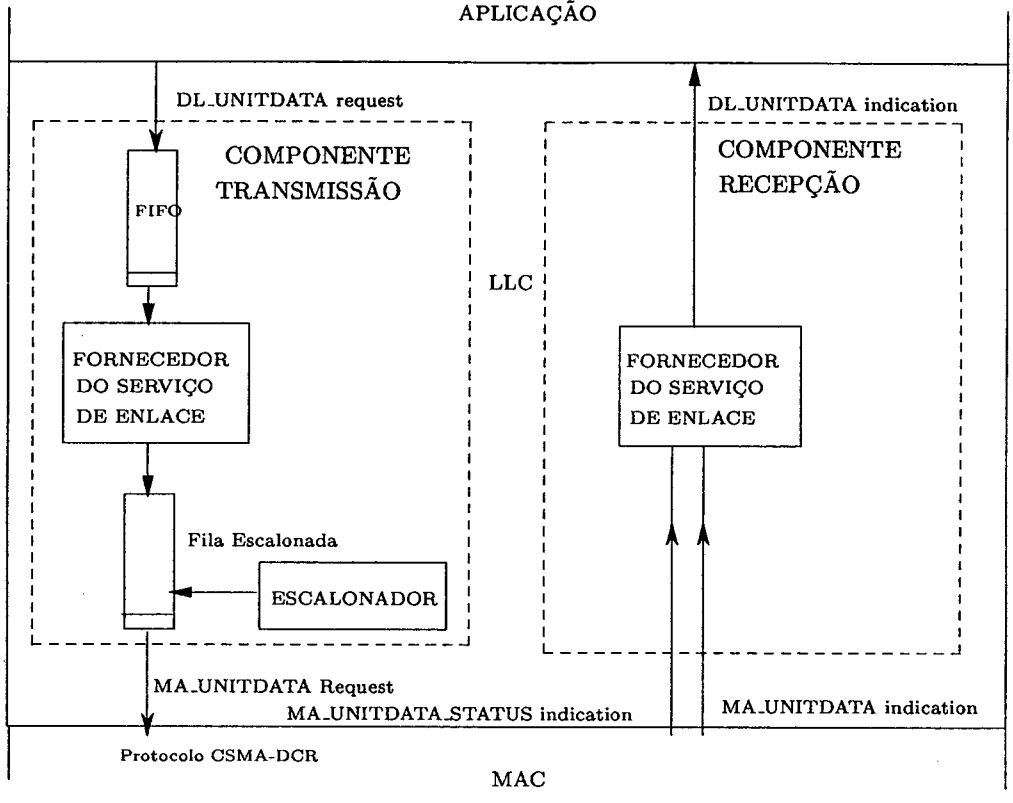

Figura 5.2: Diagrama em blocos da entidade LLC

Na seqüência é apresentado uma especificação dos serviços de enlace segundo o padrão, localizando o serviço utilizado nesta proposta. Logo depois, se detalha o protocolo de enlace envolvido para este tipo de serviço, assim como também se especifica o funcionamento do escalonador, sendo este o único elemento que se adiciona ao padrão tomado como referência.

5.2.1 Os tipos de serviços para uma camada LLC.

Para a camada LLC se definem, segundo o padrão ANSI/IEEE 802.2 (ISO 8802-2), três tipos de operações ou serviços, eles são, serviço sem conexão e sem reconhecimento(operação tipo 1), serviço com conexão(operação tipo 2) e serviço sem conexão e com reconhecimento(operação tipo 3). Detalhando eles seria:

- tipo 1: é possível trocar PDUs(Protocol Data Units) entre LLCs sem a necessidade de estabelecer uma conexão e os PDUs não precisam ser reconhecidos. Neste tipo de operação não se realiza controle de fluxo, nem recuperação de erros.

Tipos de operação	*Classes de LLC*			
	I	II	III	IV
1	x	x	x	x
2		x		x
3			x	x

Tabela 5.1: Classes de LLC

- tipo 2: é necessário o estabelecimento de uma conexão para trocar PDUs entre LLCs. O ciclo normal de uma comunicação seria, primeiro a transferência de PDUs contendo informação, desde o LLC emissor ao LLC destino, seguido do reconhecimento no sentido contrário. Neste tipo de operação existe seqüenciamento, controle de fluxo e recuperação de erros.

- tipo 3: neste tipo de operação é possível trocar PDUs entre as entidades LLCs sem necessidade do estabelecimento de uma conexão, porém a informação transmitida é reconhecida. Embora não seja estabelecida uma conexão para transmissão dos dados, existe seqüenciamento e recuperação de erros.

A partir dos tipos de operação definidos, o padrão ANSI/IEEE 802.2 também define quatro classes distintas de LLC. No sentido de associar a cada classe a possibilidade de fornecer vários tipos de operação ou serviços simultaneamente. A Classe I que deve prover operação do tipo 1; a Classe II oferece operações tipo 1 e tipo 2; a Classe III suporta operação tipo 1 e tipo 3 e por ultimo a Classe IV suporta os tipos de operação 1,2 e 3(tabela 5.1).

Uma vez definidas as alternativas de operações de uma camada LLC segundo o padrão tomado como referência nesta proposta, temos que como o presente trabalho especifica uma camada LLC para um serviço sem conexão e sem reconhecimento, se corresponde com a operação tipo 1 definida no padrão. Observe na tabela 5.1 como este tipo de operação deve ser suportadas pelas quatro classes de LLC definidas no padrão.

Para uma operação tipo 1, serviço não confirmado, se utilizam duas primitivas: *service.REQUEST* (invocada pelo solicitante), e *service.INDICATION* (entregue ao destinatário pelo fornecedor) [SLC95].

As especificações dos serviços são dadas em forma de primitivas, que representam uma forma abstrata do intercâmbio de informação entre as camadas, em função do serviço identificado [fSEC94]. A continuação apresentaremos as primitivas de serviço para as duas interfaces.

5.2.1.1 Especificação do serviço - Interface Camada Aplicação/Camada LLC

A interface entre a camada de aplicação e a camada LLC se apresenta na forma de duas primitivas. Trata-se das seguintes primitivas:

- DL-UNITDATA request
- DL-UNITDATA indication

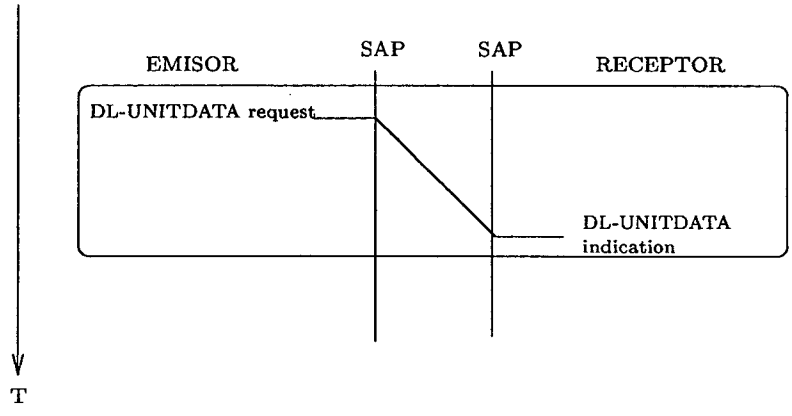

Figura 5.3: Diagrama de seqüência de tempo para as primitivas da interface entre aplicação e enlace

Na figura 5.2 se observa que a primitiva de requisição do serviço opera no componente de transmissão da estação emissora, e a primitiva de indicação do serviço opera no componente de recepção da estação receptora. A figura 5.3 mostra a seqüência delas no tempo.

1. Primitiva: DL-UNITDATA request

   A função desta primitiva é a de solicitar um serviço sem conexão e sem reconhecimento para transferência de dados. É gerada na camada de aplicação e passada para a camada LLC, para solicitar a transmissão de uma unidade de dados do serviço de enlace (LSDU - "link service data unit") para uma ou mais LSAPs("Link layer Service Access Point) remotos, utilizando procedimentos sem conexão e sem reconhecimento.

   Na recepção desta primitiva, a camada LLC fragmenta, de ser necessário, a informação que se deseja transmitir de acordo ao tamanho máximo de quadro permitido pelo protocolo MAC utilizado, que no caso é de 1518 bytes. É importante destacar que este tamanho de quadro inclui, os dados que se querem transmitir, os endereços fonte e destino dos LSAPs envolvidos e as restrições temporais da mensagem, especificadas como um deadline encapsulado no parâmetro *dados*. Logo depois, de

acordo ao deadline, se realiza o escalonamento da LSDU na fila de transmissão, para ser enviada a sua entidade par.

A semântica desta primitiva é da forma:

DL-UNITDATA request(
endereço_LSAP_fonte
endereço_LSAP_destino
dados
)

- Os parâmetros *endereço_LSAP_fonte* e *endereço_LSAP_destino* especificam os endereços dos LSAPs envolvidos na transferência de dados. O *endereço_LSAP_destino* pode especificar um endereço individual ou de grupo.

- O parâmetro *dados* especifica a LSDU a ser transferida pela entidade de enlace. Como o padrão ANSI/IEEE 802.2 (ISO 8802-2) não apresenta um campo exclusivo para especificar as restrições temporais, foi atribuido para este fim os primeiros dois bytes do campo destinado aos dados. Estas restrições temporais são utilizadas para realizar a ordenação na fila de transmissão, mas não é uma informação que seja transferida para a/as entidade/s par remota/s.

2. Primitiva: DL-UNITDATA indication

A função desta primitiva é a de indicar um serviço sem conexão e sem reconhecimento para a transferência de dados entre duas entidades pares. Esta primitiva é passada da camada LLC para a camada de aplicação para indicar a chegada de uma LSDU da entidade remota especificada.

Com a recepção desta primitiva, a camada de aplicação deve notificar ao processo de aplicação da chegada de informação.

A semântica desta primitiva é da forma:

DL-UNITDATA indication(
endereço_LSAP_fonte
endereço_LSAP_destino
dados
)

As especificações dos parâmetros *endereço_LSAP_fonte* e *endereço_LSAP_destino* são similares que da primitiva DL-UNITDATA request, a diferença está em que o parâmetro *dados* somente especifica a LSDU recebida.

5.2.1.2 Especificação do serviço - Interface Camada LLC/Camada MAC

A interface entre a camada LLC e a camada MAC se apresenta na forma de três primitivas, são utilizadas pela camada LLC para solicitar os serviços providos pela camada MAC. Estas primitivas são as utilizadas na interface com a camada MAC para todas as classes de LLC definidas no padrão ANSI/IEEE 802.2(ISO 8802-2). Elas são:

- MA-UNITDATA request
- MA-UNITDATA indication
- MA-UNITDATA-STATUS indication

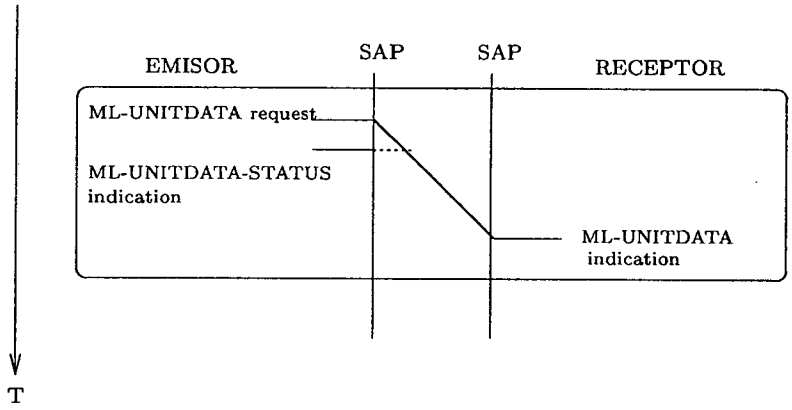

Figura 5.4: Diagrama de seqüência de tempo para as primitivas da interface entre enlace e MAC

Na figura 5.2 se observa que a primitiva de requisição do serviço opera no componente de transmissão da estação emissora, e como primitivas de indicação temos dois, uma que opera na estação emissora indicando o estado da entidade MAC local e outra que opera no componente de recepção da estação receptora. A figura 5.4 mostra a seqüência delas no tempo. A continuação se descrevem ditas primitivas.

1. Primitiva: MA-UNITDATA request

   A função desta primitiva é solicitar a transferência de uma MSDU("MAC Service Data Unit") da entidade LLC local para uma o mais entidades LLC remotas. Esta primitiva é gerada pela camada LLC e passada para a camada MAC, para requisitar a transferência de um MSDU para um ou mais entidades remotas.

Com a recepção desta primitiva a camada MAC adiciona todos os campos MAC(DA,SA) necessários para completar a informação que formam uma MSDU. Uma vez que a interface que implementa o protocolo MAC adquiere o direito ao canal de comunicação, a MSDU é enviada para um ou mais entidades remotas.

A semântica desta primitiva é da forma:

MA-UNITDATA request(
endereço_MAC_fonte
endereço_MAC_destino
dados
classe_de_serviço
)

- Os parâmetros *endereço_MAC_fonte* e *endereço_MAC_destino* especificam os endereços das entidades MAC local e remota respectivamente, envolvidas na transferência de dados. O *endereço_MAC_destino* pode especificar um endereço individual ou de grupo.

- O parâmetro *dados* especifica a MSDU a ser transferida pela entidade MAC.

- Por ultimo o parâmetro *classe_de_serviço* especifica a classe de serviço requisitada pela camada LLC.

2. Primitiva: MA-UNITDATA indication

Esta primitiva indica a transferência de uma MSDU de uma entidade MAC para uma entidade LLC, ou entidades no caso de endereços de grupos. Na ausência de erro, o conteúdo do parâmetro *dados* é igual ao parâmetro *dados* da primitiva MA-UNITDATA request que originou a indicação. Esta primitiva é gerada pela camada MAC e passada para a camada LLC, indicando a chegada de um quadro à entidade MAC local.

Com a recepção desta primitiva, se indica a chegada de informação na forma de uma MSDU que passará à camada de aplicação para ser processada. Se na estação emissora foi necessário a fragmentação da informação, com a chegada desta primitiva na estação receptora deve-se realizar o contrário, integrando a informação dos diferentes quadros.

A semântica desta primitiva é da forma:

MA-UNITDATA indication(
endereço_MAC_fonte
endereço_MAC_destino
dados
estado_recepção
classe_de_serviço
)

- O parâmetro *endereço_MAC_fonte* deve especificar um endereço individual, presente no campo SA do quadro recebido. O *endereço_MAC_destino* pode especificar um endereço individual ou de grupo, presente no campo DA do quadro recebido.

- O parâmetro *dados* corresponde à MSDU recebida pela entidade MAC.

- O parâmetro *estado_recepção* indica o sucesso ou falha do quadro que chega.

- Por ultimo o parâmetro *classe_de_serviço* que especifica a classe de serviço requerida.

3. Primitiva: MA-UNITDATA-STATUS indication

Esta primitiva é associada à primitiva MA-UNITDATA request. É uma indicação do estado da entidade MAC local. É gerada pela camada MAC e passada para a camada LLC, para indicar o estado da primitiva MA-UNITDATA request associada.

Com a chegada desta primitiva, em função do valor do parâmetro *estado_transmissão*, se envia novamente ou não, a primitiva MA-UNITDATA request.

A semântica desta primitiva é da forma:

MA-UNITDATA-STATUS indication(
endereço_MAC_fonte
endereço_MAC_destino
estado_transmissão
classe_de_serviço_fornecida
)

- Os parâmetros *endereço_MAC_fonte* e *endereço_MAC_destino* especificam o mesmo que na primitiva MA-UNITDATA indication.

- O parâmetro *estado_transmissão* indica o sucesso ou falha do quadro enviado pela entidade MAC local, quem verifica o bit que indica o status da transmissão realizada.

- O parâmetro *classe_de_serviço* se refere ao parâmetro usado na primitiva precedente (MA-UNITDATA request).

O uso desta primitiva é uma alternativa na ajuda na falta de mecanismos de controle de erro existente na camada LLC para o serviço sem conexão e sem reconhecimento, indica o sucesso o falha do MA-UNITDATA request anterior. Esta primitiva pode ser usada para retornar o estado à entidade LLC local.

5.2.2 Aspectos relativos ao protocolo de enlace

No anexo se apresenta o detalhe da estrutura da LLC PDU, mas de forma geral pode-se dizer que está composta de 4 campos. O campo *DSAP* que indica o ponto de acesso da entidade receptora, o campo *SSAP* que indica o ponto de acesso da entidade emissora, o campo de controle que indica o tipo de unidade de dados de protocolo, e por ultimo o campo de informação. Na tabela 5.2 se mostra o formato da LLC PDU (Protocol Data Unit):

DSAP	*SSAP*	*Controle*	*Information*
8 bits	8 bits	8 bits	N*8 bits

Tabela 5.2: Formato de uma LLC PDU

No anexo se especificam os formatos dos campos de endereçamento e do campo de controle, entretanto é importante descrever alguns aspectos por sua relevância na implementação do protocolo de enlace entre as entidades LLC.

5.2.2.1 Comandos e Respostas

Este trabalho implementa um serviço sem conexão e sem reconhecimento, o que se corresponde com uma operação tipo 1. Trata-se de um serviço para aplicações que não requerem reconhecimentos dos dados nem controle de fluxo. O conjunto de comandos e respostas PDUs suportados por esta classe de serviço são os que se observam na tabela 5.3.

- UI

  O quadro *UI* só existe como comando. A recepção do comando *UI* não é reconhecida nem seqüenciada, portanto a informação pode ser perdida no caso de acontecer algum tipo de erro na transmissão, ou que o receptor este ocupado no momento de

Comando	*Resposta*
UI	
XID	XID
TEST	TEST

Tabela 5.3: Comandos e Respostas para operação tipo 1

envio do comando *UI*. Seu uso pressupõe que a probabilidade de perda da informação seja muito pequena.

- XID

  Este comando é opcional na implementação, mas é obrigatoria a resposta à recepção dele. O comando/resposta XID é utilizado para procedimentos de identificação de outras entidades LLC, assim como também para determinar os tipos de serviços suportados pelas entidades LLC remotas.

  Possíveis usos do comando XID :

  - O comando XID com endereços DSAP e SSAP nulos solicita a resposta de alguma estação.
  - O comando XID com um endereço DSAP de grupo pode ser utilizado para determinar os membros de um grupo.
  - O comando XID com um endereço DSAP global pode ser usado para identificar todas as estações ativas.
  - Pode ser utilizado para determinar o tipos de serviços ou classes de LLC dos SAP especificados nos endereços. Para isto tem que se examinar o campo de informação do XID.

- TEST

  O comando/resposta TEST pode ser utilizado para os testes de "loopback". Usado para fazer com que o LLC destinatário envie uma resposta TEST tão breve quanto possível. Quando a informação recebida no campo de informação pela entidade LLC emissora, é a mesma ao que ela enviou, pode-se concluir que o teste concluiu satisfatoriamente. Da mesma forma que com XID, este comando é opcional na implementação, mas é obrigatoria a resposta à recepção de dele.

5.2.3 Componente de transmissão

Na figura 5.2 se observa que na entrada do componente de transmissão, existe uma fila com ordem FIFO, esta fila armazena as primitivas de requisição de serviço da camada LLC, elas se encontram na forma de uma LSDU. Neste trabalho se especificam somente as primitivas para uma operação tipo 1, serviço sem conexão e sem reconhecimento.

Estas primitivas são entregues ao fornecedor do serviço de enlace, quem faz a fragmentação(caso seja necessária) da mensagem e o cálculo dos deadlines dos pacotes. Para o calculo dos deadlines dos pacotes, realiza-se a extração e armazenamento dos dois primeiros bytes de dados do parâmetro *dados* da primitiva de requisição de serviço da camada de enlace, nesses bytes se encontram as restrições temporais da mensagem, expressadas como um deadline. Obtidos os pacotes e calculado seus deadlines, se formatam como uma unidade de dados do protocolo de enlace para ser colocados na fila de transmissão, onde são ordenados pelo escalonador de acordo à política de atribuição de prioridades, no caso EDF("earliest deadline first").

A continuação se descreve como é realizada a fragmentação das mensagens e como é o funcionamento do escalonador preemptivo utilizado nesta proposta.

5.2.3.1 Fragmentação

Se o tamanho da mensagem é maior que o tamanho máximo em bits reservado para o envio de dados do quadro definido pelo protocolo MAC, a mensagem deve ser fragmentada em N pacotes. O protocolo utilizado nesta proposta, o CSMA-DCR, apresenta como máximo tamanho de pacote 1518 bytes, a continuação se apresenta o formato do quadro de dados do protocolo:

7 bytes	*Preâmbulo*
1 byte	*Delimitador de início*
6 bytes	*Endereço Destino*
6 bytes	*Endereço Fonte*
2 byte	*Comprimento*
70 - 1492	*Dados* *PAD*
4 bytes	*Checksum*

Tabela 5.4: Formato do quadro MAC

Do formato 5.4 se deduz que existe uma informação adicional(preâmbulo, endereços de destino e fonte, delimitador de início, etc) característica do protocolo utilizado na camada MAC. Os bytes que ocupam ditos campos devem ser descontados do tamanho

máximo do pacote, para desta forma poder calcular o número de bytes disponíveis para transmitir dados, no caso restariam como máximo 1492 bytes para ser utilizados para a transmissão de dados.

Entretanto, ainda temos que definir como encapsular as restrições temporais e o sequenciamento da mensagem fragmentada. Devido a que está-se utilizando uma operação do tipo 1, os quadros suportados são não numerados, portanto, é necessário especificar de alguma forma o começo e fim de uma mensagem na seqüência de pacotes desprendidos da fragmentação, para poder realizar a blocagem dos pacotes na estação receptora desta informação.

Para não modificar o padrão, a proposta é :

- dentro do campo *dados* da LLC PDU atribuir os quinze bits menos significativos do quarto e quinto byte para a especificação do número da sequência do pacote.

- dentro do campo *dados* da LLC PDU atribuir um bit mais significativo do quarto byte para a especificação do fim de mensagem.

- dentro do campo *dados* atribuir o terceiro e quarto byte para a especificação do deadline do pacote.

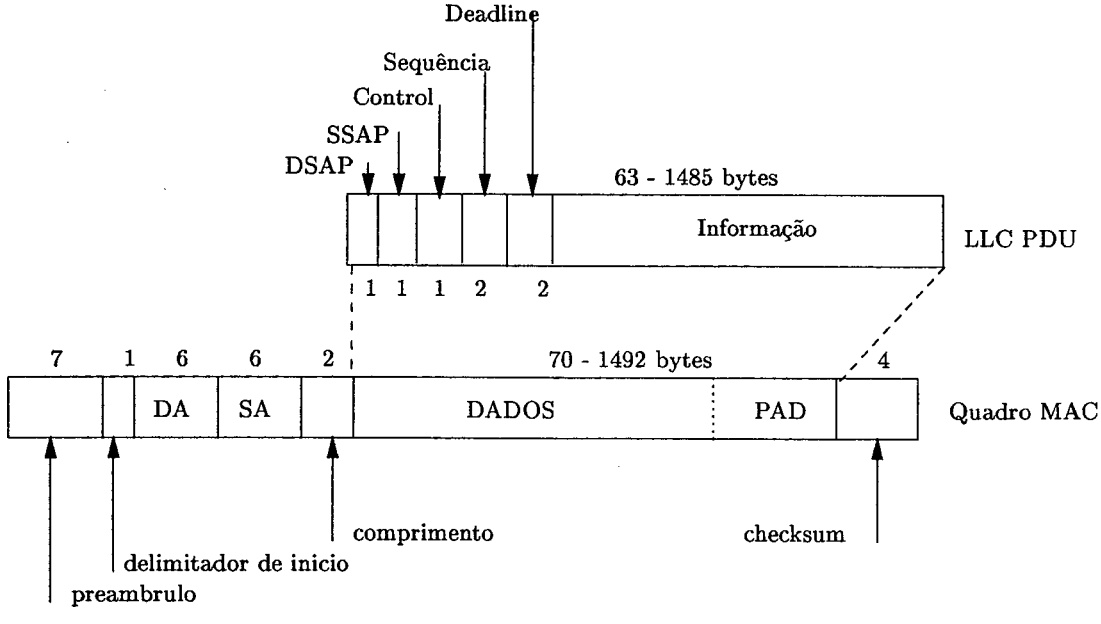

Figura 5.5: Encapsulamento no quadro MAC

No formato padrão de uma LLC PDU existem 3 bytes próprios que devem ser considerados, o campo DSAP, o campo SSAP e o campo de controle.

Considerando os bytes acima temos que, para transmissão de dados restariam 1485 bytes, sendo este o número de bytes que deve ser tomado como referência para determinar se a mensagem recebida da camada de aplicação necessita ser fragmentada em pacotes.

Na figura 5.5 se ilustra o explicado anteriormente.

5.2.3.2 Escalonador

A ordenação na fila de transmissão é realizada por um escalonador preemptivo ativado por eventos, utilizando o EDF como política de atribuição de prioridades. O cálculo das prioridades para realizar a ordenação na fila de transmissão, se faz utilizando a técnica proposta em [MNS96], e apresentada neste trabalho na seção 3.6. Como o escalonamento se faz com uma visão local, não é necessário o sincronismo de relógio entre as diferentes estações que conformam a rede local.

- **Cálculo do deadline de pacote**

  Definindo C_{dados} como o tempo de transmissão de 1485 bytes e C_i o tempo de transmissão da mensagem m_i, temos que:

$$se \quad C_i \geq C_{dados} \quad então \quad N = \left\lceil \frac{C_i}{C_{dados}} \right\rceil$$

  Calculado o número total de pacotes N que se desprende da fragmentação, é necessário calcular o deadline do pacote.

  Os deadlines de cada pacote podem ser calculados de duas formas, uma primeira seria definindo o deadline de cada pacote como igual ao tempo de transmissão de um pacote de tamanho máximo, a desvantagem é que apresenta uma folga muito pequena, existindo grandes possibilidades de que o pacote perca o deadline.

  Outra forma, sendo esta à adotada nesta proposta, é a distribuição da folga $d_i - C_i$ entre os N pacotes da seguinte forma:

$$d_{pacote} = C_{dados} + \frac{d_i - C_i}{N}$$

  É importante destacar que existe um custo associado à granularidade do relógio do sistema que não consideramos neste trabalho. Por exemplo, no caso que a diferença de deadlines entre duas mensagens seja menor que a granularidade do relógio, ambas mensagens serão tratadas e analisadas como possuindo o mesmo deadline, quando na situação real são mensagens que deveriam ser tratadas com prioridades diferentes.

  Obtido o deadline de pacote, o fornecedor de serviço de enlace formata o pacote como uma LLC PDU, introduzindo o deadline como um inteiro no terceiro e quarto

byte da LLC PDU. Finalmente o pacote é enfileirado na fila de transmissão, a qual é ordenada em função do deadline de pacote.

5.2.4 Componente de recepção

Comparando com o componente de transmissão, o componente de recepção é mais simples. Isto se explicaria porque a camada LLC especificada utiliza um serviço sem conexão e sem reconhecimento, portanto as primitivas envolvidas na recepção não realizam controle de erro e de fluxo, simplificando em grande medida o protocolo de enlace.

Outro aspecto que simplifica o componente de recepção é que o sistema de comunicação analisado, dentro da classificação de sistemas tempo real apresentada no capítulo 3, se enquadrada como um sistema com garantia no projeto, ou seja, que com uma carga conhecida a priori, a análise de escalonabilidade realizada na fase de projeto, se garante que as restrições temporais das mensagens são respeitadas, portanto na recepção não é necessário analisar o deadline da mensagem.

A única função que resta para o componente de recepção é a blocagem dos pacotes recebidos da camada MAC, obtendo desta forma a mensagem a qual é enviada para a camada de aplicação. Para realizar a blocagem se verifica o byte que indica a sequência dos quadros que formam a mensagem, assim como também o bit que define o fim da mensagem fragmentada.

5.2.5 Diagrama de estado da entidade de enlace

Analisando com uma visão abstrata a entidade de enlace pode estar inativa, quando não se requer que desenvolva nenhum serviço, ou ativa, quando se lhe faz alguma requisição de serviço de enlace. O diagrama de estado abstrato se mostra na figura 5.6.

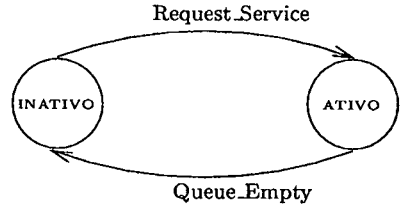

Figura 5.6: Diagrama abstrato da entidade LLC

Expandindo o diagrama anterior, no sentido de expandir os eventos que ativam um estado ou outro e expandir o estado ativo da entidade LLC temos a figura 5.7.

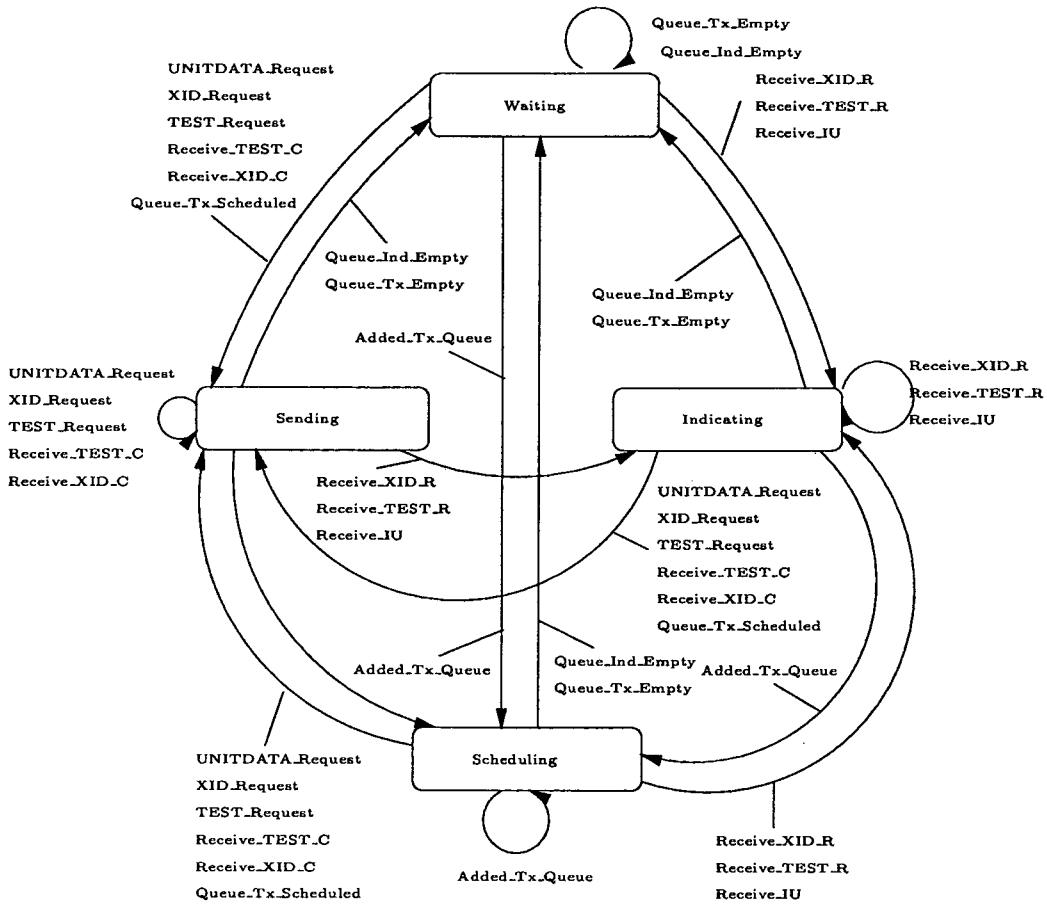

Figura 5.7: Diagrama de estado da entidade LLC

5.2.6 Tabela de transição de estados da entidade LLC

Na tabela 5.5 se apresenta a tabela de transição de estados da entidade LLC:

Estado Atual	Evento	Ação	Prox. Estado
Waiting	UNITDATA_Request	Send_UI	Sending
	XID_Request	Send_XID_C	Sending
	TEST_Request	Send_TEST_C	Sending
	Receive_TEST_C	Send_TEST_R	Sending
	Receive_XID_C	Send_XID_R	Sending
	Receive_UI	UNITDATA_Indication	Indicating
	Receive_TEST_R	TEST_Indication	Indicating
	Receive_XID_R	XID_Indication	Indicating
	Queue_Scheduled	Send_Next	Sending
	Queue_Tx_Empty	Sem_ação	Waiting
	Queue_Ind_Empty	Sem_ação	Waiting
	Added_Tx_Queue	Escalonar	Scheduling
Sending	UNITDATA_Request	Send_UI	Sending
	XID_Request	Send_XID_C	Sending
	TEST_Request	Send_TEST_C	Sending

Continua na página seguinte

Continuação da página anterior			
	Receive_TEST_C	Send_TEST_R	Sending
	Receive_XID_C	Send_XID_R	Sending
	Receive_UI	UNITDATA_Indication	Indicating
	Receive_TEST_R	TEST_Indication	Indicating
	Receive_XID_R	XID_Indication	Indicating
	Queue_Scheduled	Send_Next	Sending
	Queue_Tx_Empty	End_Procedure	Waiting
	Queue_Ind_Empty	End_Procedure	Waiting
	Added_Tx_Queue	Escalonar	Scheduling
Indicating	UNITDATA_Request	Send_UI	Sending
	XID_Request	Send_XID_C	Sending
	TEST_Request	Send_TEST_C	Sending
	Receive_TEST_C	Send_TEST_R	Sending
	Receive_XID_C	Send_XID_R	Sending
	Receive_UI	UNITDATA_Indication	Indicating
	Receive_TEST_R	TEST_Indication	Indicating
	Receive_XID_R	XID_Indication	Indicating
	Queue_Scheduled	Send_Next	Sending
	Queue_Tx_Empty	End_Procedure	Waiting
	Queue_Ind_Empty	End_Procedure	Waiting
	Added_Tx_Queue	Escalonar	Scheduling
Scheduling	UNITDATA_Request	Send_UI	Sending
	XID_Request	Send_XID_C	Sending
	TEST_Request	Send_TEST_C	Sending
	Receive_TEST_C	Send_TEST_R	Sending
	Receive_XID_C	Send_XID_R	Sending
	Receive_UI	UNITDATA_Indication	Indicating
	Receive_TEST_R	TEST_Indication	Indicating
	Receive_XID_R	XID_Indication	Indicating
	Queue_Scheduled	Send_Next	Sending
	Queue_Tx_Empty	End_Procedure	Waiting
	Queue_Ind_Empty	End_Procedure	Waiting
	Added_Tx_Queue	Escalonar	Scheduling

Tabela 5.5: Tabela de transição de estados da entidade LLC

5.2.6.1 Descrição dos estados da entidade LLC

A entidade LLC quando ativa, pode-se encontrar esperando a chegada de uma requisição de serviço da camada de aplicação ou chegada de um comando XID_C ou TEST_C de uma entidade LLC remota (estado "Waiting"). Pode estar enviando para a camada MAC, os pacotes formatados que estão enfileirados na fila de transmissão(estado "Sending"). Pode-se encontrar indicando para a camada de aplicação da chegada de algum pacote da camada MAC(estado "Indicating") ou pode-se encontrar ordenando a fila de transmissão de acordo à política de atribuição de prioridades, no caso EDF(estado "Scheduling"):

- Waiting: espera a requisição de seus serviços com a chegada de uma SDU ("Service Data Unit") da camada de aplicação ou a recepção de algum pacote no formato de uma LLC PDU da camada MAC.
- Sending: As mensagens são enviadas para a camada MAC no formato de uma LLC PDU.
- Indicating: indica para a camada de aplicação da chegada de uma LLC PDU da camada MAC.
- Scheduling: insere os pacotes na posição adequada na fila de transmissão em função da prioridade.

5.2.6.2 Descrição dos eventos da entidade LLC

A continuação se detalham os eventos que ativam os diferentes estados da entidade LLC:

- UNITDATA_Request: um usuário SAP requisita o envio de uma unidade de dados para um ou mais LLCs remotos, via um UI PDU.
- XID_Request: um usuário SAP requisita que seja enviado um comando XID para um o mais LLCs remotos, iniciando-se desta forma um procedimento XID.
- TEST_Request: um usuário SAP ha requisitado que seja enviado um comando TEST para um o mais LLCs remotos, iniciando-se desta forma um procedimento TEST.
- Receive_TEST_C: se recebe de um LLC remoto um comando TEST.
- Receive_XID_C: se recebe de um LLC remoto um comando XID.
- Receive_TEST_R: a entidade LLC local recebeu uma resposta TEST de um LLC remoto.
- Receive_XID_R: a entidade local recebeu uma resposta XID de um LLC remoto.
- Receive_UI: a entidade local recebeu um UI PDU de um LLC remoto.
- Queue_Scheduled: os pacotes da fila de transmissão foram ordenados em função de suas restrições temporais.

- Queue_Tx_Empty: indica que não há mensagens na fila de transmissão esperando para ser enviadas para a camada MAC.
- Queue_Ind_Empty: indica que não há mensagens na fila de indicação esperando para ser enviadas para a camada aplicação.
- Added_Tx_Queue: indica que foi adicionado um novo pacote na fila de transmissão.

5.2.6.3 Descrição das ações da entidade LLC

Quando ativado um evento, inicia-se uma ação, a continuação se descrevem as ações associadas aos eventos:

- Send_UI: uma UI PDU é enviada para um ou mais LLCs remoto em resposta a uma requisição de um usuário para enviar uma SDU.
- Send_XID_C: um comando XID é enviado para um ou mais LLCs remoto em resposta a uma requisição de um usuário para identificar outros LLCs.
- Send_TEST_C: um comando TEST é enviado para um ou mais LLCs remoto em resposta a uma requisição de um usuário para testar outros LLCs.
- Send_TEST_R: envia uma resposta TEST para a camada MAC produto da chegada de um comando TEST de um LLC remoto.
- Send_XID_R: envia uma resposta XID para a camada MAC produto da chegada de um comando XID de um LLC remoto.
- UNITDATA_Indication: se envia para a camada de aplicação uma indicação da chegada de uma UI PDU.
- XID_Indication: se envia para a camada de aplicação uma indicação da chegada de uma resposta XID de uma entidade LLC remota.
- TEST_Indication: se envia para a camada de aplicação uma indicação da chegada de uma resposta TEST de uma entidade LLC remota.
- Escalonar: se passa a escalonar a fila de transmissão.
- Send_Next: como a fila esta ordenada pode-se enviar o pacote mais prioritário dela.
- End_Procedure: a entidade LLC concluiu as requisições com sucesso passando a esperar outra requisição de serviço.

5.2.7 Procedimentos

Nas figuras 5.8, 5.9 e 5.10 se descrevem os procedimentos possíveis para o protocolo de enlace especificado neste trabalho, eles são, o envio de uma unidade de dados para uma ou mais entidades de enlace remotas(Procedimento UI), o pedido identificação de uma ou mais entidades LLC(Procedimento XID), este procedimento também pode ser utilizado para determinar os tipos de serviços suportados por uma entidade LLC e por último o procedimento de teste entre entidades de enlace(Procedimento TEST).

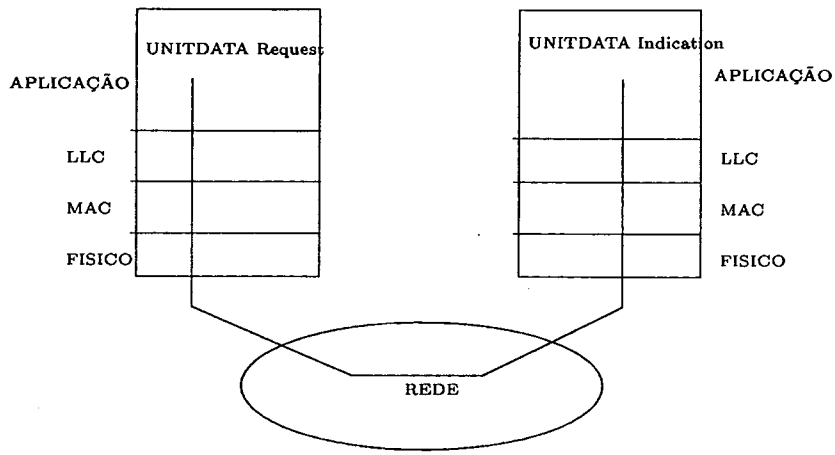

Figura 5.8: Procedimento UI

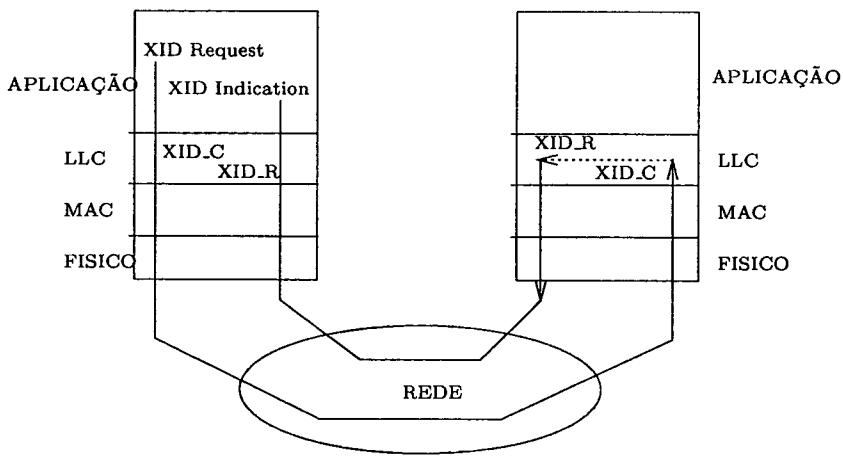

Figura 5.9: Procedimento de identificação: XID

5.3 Estudo sobre implementação

Na continuação se descrevem aspectos relativos à implementação da camada de enlace especificada nesta proposta, como são, do ponto de vista do *hardware*, as características

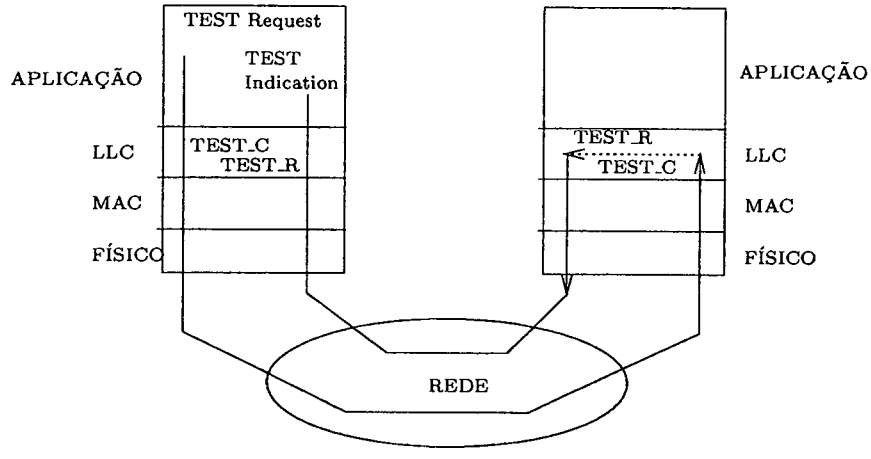

Figura 5.10: Procedimento de teste: TEST

da interface que implementa o protocolo MAC e do ponto de vista do *software* as características do sistema que deve ser desenvolvido. A proposta de implementação será abordada por camadas, ou seja, primeiro se apresentará o suporte de software para a camada LLC e depois o suporte tanto de software como de hardware para a camada MAC.

5.4 Definição do suporte de software da camada LLC

O suporte do software para a camada LLC que se especifica neste trabalho deve considerar a existência de um núcleo tempo real que permita a execução simultânea, paralela ou concorrente de vários processos. Este executivo tempo real deve assegurar e gerenciar os processos, através de mecanismos de escalonamento que garantem a execução das tarefas mais prioritárias, a intercomunicação entre processos e o gerenciamento dos eventos que ativam os processos. Para o estudo de implementação se utilizou o núcleo de sistema operacional distribuído com mecanismos para tempo real proposto por Nacamura em [Jún88].

A chegada de uma requisicão de serviço será vista pelo sistema de suporte como um evento que dispara um processo servidor ("demoen"). Devem existir alguns processos encarregados dos serviços LLC que implementa a máquina de estado da entidade LLC.

5.4.1 Núcleo Tempo Real

O Núcleo Tempo Real ou Executivo Tempo Real é o responsável pela implantação do ambiente multitarefas permitindo a execução da aplicação distribuída. É um programa

distribuído composto de vários módulos ou processos(tarefas) que se executam de forma concorrente e cooperando entre si para a realização de uma atividade, está constituído por um conjunto de primitivas acessadas em grande parte por tarefas em tempo de execução.

Como funções do núcleo temos:

- assegurar o gerenciamento das tarefas, através de mecanismos de escalonamento que garantem a execução das tarefas mais prioritárias.
- assegurar a intercomunicação e sincronização entre tarefas, inclusive o gerenciamento dos recursos do sistema e comunicação remota.
- assegurar o gerenciamento dos eventos externos, tratamento das interrupcoes.
- assegurar o gerenciamento do tempo.

No presente trabalho utiliza-se a abordagem clássica para o projeto do software da decomposição modular.Os servicos fornecidos por uma camada são descompostos e implementados através de módulos, os quais gerenciam um conjunto de recursos físicos ou lógicos. As cooperações entre módulos, assim como as requisições de servico de uma camada se realizam através de interfaces, portanto existem dois tipos de interfaces, uma entre camadas e outra entre módulos cooperantes de uma dada camada. No caso do presente trabalho, associa-se um módulo a sub-camada LLC e outro módulo a sub-camada MAC.

A definição de entidades que expressem modularidade, concorrência e distribuição facilita a elaboração da aplicação distribuída. Em tempo de execução, a noção de modularidade é substituída pela noção de concorrência. As tarefas ou processos são criados durante a instanciação de um móduloo e representam a entidade elementar de concorrência. Se considera, que um programa distribuído é um conjunto de tarefas cooperantes em execução, competindo(ou compartilhando) pelos recursos do sistema, onde a cooperação entre tarefas realiza-se por troca de mensagens, sendo esta a unidade elementar para o transporte de informação.

5.4.1.1 Portos

A comunicação entre tarefas se realiza por meio de transmissões síncronas e assíncronas através de interfaces bem definidas denominadas "porto", ou seja as tarefas enviam mensagens através de portos de saída e recebem via porto de entrada. Em tempo de configuração, são conectados os portos de saída aos portos de entrada, formando canais de comunicação e sincronismo entre tarefas.

O núcleo tempo real fornece um conjunto de primitivas (ligar, religar e desligar porto), que permitem o estabelecimento dos canais de comunicação e sincronismo entre módulos (na configuração) e entre tarefas (na instanciação do tipo módulo). No modelo de troca de mensagens, a noção de porto é importante porque soluciona os problemas de endereçamento e tratamento de mensagens associadas as tarefas e separa a programação de componentes da programação da configuração do sistema, ou seja, módulos ou tarefas só referenciam a objetos locais, seus portos.

5.4.1.2 Primitivas de comunicação e sincronismo

Partindo das conexões lógicas de dois ou mais portos, as mensagens endereçadas à um porto de saída podem ser recebidas num porto de entrada. Esta conexão define um canal de comunicação que é representado e manipulado pelas funções do núcleo através de uma estrutura de ligação, onde as primitivas de comunicação definidas permitem os endereçamentos: um-para-um, vários-para-um e um-para-vários (somente em transmissões assíncronas).

As tarefas enviam mensagens para transferir dados ou sinais para uma ou mais tarefas, ou pela necessidade de sincronização com uma ou mais tarefas, ou porque solicita o serviço de uma tarefa.

Existem tres tipos básicos de primitivas de envio, o tipo envio não bloqueante, o envio síncrono aguardando recepção e por último o envio síncrono aguardando resposta. As duas primeiras representam canais de comunicação unidirecional e a última um canal de comunicação bidirecional. Como primitivas de recepção temos as não bloqueantes e as bloqueantes:

- envio síncrono aguardando recepção: o processo que envia a mensagem fica bloqueado até que a mensagem seja recebida pelo processo destino, ou seja, que além da transmissão de dados, se produz uma sincronização entre o emissor e o receptor da mensagem. Este sincronismo está condicionado somente a recepção da mensagem. Não é necessário o armazenamento da mensagem porque esta é transferida somente se o receptor estiver pronto. Mecanismos devem ser implementados para evitar bloqueios por tempos indefinidos da tarefa emissora, um exemplo é a técnica do "timeout".

- envio síncrono aguardando resposta: além da sincronização entre a tarefa emissora e a tarefa receptora, este sincronismo se mantém até a chegada de uma mensagem resposta à tarefa emissora. Não existe a necessidade de um buffer, se no momento do envio a tarefa receptora não se encontra em condições de receber a mensagem,

a tarefa emissora é bloqueada até receber a mensagem resposta. Também é necessária a implementação de mecanismos que evitem os bloqueios indefinidos da tarefa emissora.

- envio não bloqueante: quando o processo emissor da mensagem continua o seu procesamento logo após o armazenamento da mensagem num buffer. Este tipo de envio não necessariamente sincroniza a tarefa emissora com a tarefa receptora, ou seja, se no instante do envio de uma mensagem a tarefa receptora não se encontra em condições de recebe-la, a mensagem é colocada num buffer até que ocorra a recepção. Disto se depreendem problemas como, que a mensagem recebida não necessariamente contém informações atuais da tarefa emissora, a limitação física dos buffers, para o qual deve-se definir o tratamento quando não tem disponibilidade de buffer, no caso optou-se por suspender a tarefa emissora até que o buffer esteja disponível.

- recepção bloqueante: esta primitiva é a mais común, porque geralmente uma tarefa esperando pela recepção de uma mensagem não tem nada a fazer além disto. A tarefa receptora é bloqueada até que uma mensagem seja colocada na fila do canal de comunicação correspondente, existindo portanto uma sincronização entre a tarefa emissora e a tarefa receptora. Para evitar a espera indefinida, é necessário um mecanismo de "timeout". Um caso particular sería a primitiva com um "timeout"igual zero, representando o caso de uma primitiva de recepção não bloqueante. Se a mensagem foi enviada utilizando uma primitiva de envio não bloqueante, é colocada num buffer até que se dê a sua recepção. Se a mensagem foi enviada utilizando uma primitiva bloqueante, a tarefa emissora ficará bloqueada até que ocorra a recepção.

5.4.1.3 Representação das tarefas

O ambiente multitarefa se obtem a partir de estruturas de dados que representam logicamente todas as tarefas do sistema. Cada tarefa possui um descritor ou bloco de controle que contém as informações relevantes sobre a tarefa. Ditos descritores de tarefas são organizados de forma facilmente manipulável, para isto se utilizam filas onde os descritores podem ser facilmente inseridos ou removidos. Existe também uma tabela central que contém apontadores para acesso as várias filas de tarefas.

Os possíveis estados de uma tarefa durante sua execução são: ativa, pronta, espera, suspensa e definida.

- ativo: é a tarefa mais prioritária entre as tarefas em estado de pronta, a tarefa está

sendo executada pelo processador.

- pronto: são as tarefas que possuem todos os recursos necessários para sua execução, exceto o processador.
- espera: necessitam de algum recurso para retornar a execução.
- suspenso: não concorrem mais ao processador, foram suspensas devido a ocorrência de algum erro de tempo de execução.
- definido: possuem todas as estruturas de dados necessárias a sua execução, mas ainda não estão concorrendo ao processador.

Associa-se uma fila a cada estado, excetuando-se os estados ativo e definido. Ditas filas se organizam de acordo com algúm critério. A ordem de ocupação do processador se estabelece através da fila de pronto.

5.4.1.4 Escalonador

O escalonador é requisitado sempre que uma tarefa tem a sua execução suspensa o bloqueada, e o processador deve ser alocado a outra tarefa. Isto ocorre no caso de uma interrupção externa que muda o estado de uma tarefa, ou da suspensão temporária de alguma tarefa produto da execução de alguma primitiva, ou pela detecção de erro, o qual impede que a tarefa prossiga a sua execução. O escalonador é ativado por eventos, seguindo uma estrategia ou política na ordenação na ocupação do processador por parte das tarefas, no caso, por prioridades. O escalonador verifica se a tarefa corrente é a mais indicada para ser executada. Se for, retorna o controle para a mesma. Caso contrário, salva o contexto da tarefa corrente no respectivo descritor ou na área de stack da mesma, selecciona a tarefa para ser executada, recuperando seu contexto e transferindo o controle para dita tarefa selecionada.

5.4.1.5 Tratamento da interrupção

Para melhorar a eficiência, em vez de integrar a manipulação de interrupções com a semântica do modelo troca de mensagens, onde a ocorrência de uma interrupção gera uma mensagem, que é endereçada a um porto, que ativa uma tarefa a qual trata o evento associado com a interrupção, optou-se por sinalizar diretamente a tarefa quando ocorre uma interrupção, ou seja que na ocorrência da interrupção a tarefa concorre ao processador em tempo mínimo. As interrupções respondem a sinais gerados exteriormente (interrupções de hardware) ou internamente (interrupções de software).

Existe uma tabela de interrupção interna ao núcleo, onde estão relacionados os vetores físicos de interrupção (vetores de interrupção do processador) com os vetores lógicos de interrupção (níveis lógicos gerenciados pelo núcleo). Existem funções internas ao núcleo, que fazem parte do gerenciamento de interrupção e existem funções que podem atender interrupções a nível de aplicação.

Na camada de enlace especificada, a recepção remota é sinalizada através de interrupção.

5.4.1.6 Temporização

O objetivo da função de temporização é prover uma base de tempo local através da atualização periódica do relógio tempo real, o qual é utilizado no gerenciamento das comunicações síncronas temporizadas, as quais utilizam o mecanismo de "timeout", além de prover uma base de tempo para tarefas periódicas, e sincronização de ativações com ações externas.

Existe uma tarefa encarregada de chamar à função que incrementa periodicamente o relógio tempo real, dita tarefa é ativada por interrupção e se atribui uma prioridade máxima. Quando ativada se verificam os "timeout"da fila de espera.

5.4.1.7 Comunicação remota

O núcleo tempo real permite a comunicação entre diferente estações de forma transparente para as tarefas aplicativas, de forma tal que não implique em distinções entre comunicação local e remota. Para lograr isto se define um módulo Servidor de Rede composto de uma tarefa servidor de rede de envio e de uma ou várias tarefas servidor de rede de recepção as quais se encontram na sub-camada MAC.

5.5 Descrição do suporte de software da camada MAC

A núcleo tempo real proposto no trabalho [Jún88] utiliza como suporte de comunicação uma rede local comercial com topologia de barramento com protocolo de acesso CSMA-CD. Tanto a interface de rede, como o driver de rede especificados naquele trabalho, não correspondem com o objetivo do presente trabalho, ou seja a especificação de uma camada de enlace para o envio de mensagens com restrições temporais, utilizando um protocolo determinista. Por isto, as tarefas servidoras de rede devem ser re-definidas de forma tal que considere a posibilidade de utilizar a interface de rede para aplicações

deterministas (utilizando o protocolo de acesso CSMA-DCR) ou probalistas (utilizando o protocolo de acesso CSMA-CD).

No presente trabalho as tarefas Servidoras de Rede são requisitadas através das primitivas adotadas pelo padrão IEEE 802.2 para a interface entre a camada LLC e a camada MAC. Estas tarefas são abordadas com maior detalhe na seção que trata da descomposição modular da camada de enlace.

5.6 Definição do suporte de hardware da camada MAC

Esta seção pretende apresentar a interface utilizada neste trabalho para implementar o protocolo MAC CSMA-DCR. Trata-se da placa de rede Dethernet 104 do fabricante APTOR/ITMI.

A arquitetura da placa Dethernet 104 pode ser genericamente descrita como sendo do tipo "mestre sem memória". O controlador lê e escreve diretamente dentro da memória do PC, numa faixa especificamente reservada.

A arquitetura da placa pode ser dividida em três partes:

- **Parte inteligente**: formada pelo coprocessador 82596SX e o controlador de barramento ISA
- **Parte de zona de Informações**: formada por
  - PROMs de Boot.
  - código de identificação da placa Ethernet: código internacional de 48 bits definido pela IEEE para cada estação, respeitando a norma 802.3. No hardware, este código é armazenado sob a forma não volátil num PLD e é acessível dentro do espaço de I/O.
  - as micro-chaves de configuração do endereço de base da zona de I/O
  - o código de identificação Factor, utilizado como o índice da estação na implementação do algoritmo DCR. Cada estação possui um código de identificação único, variando entre 0 e 255. Este código se configura por "switch"e portanto pode ser atribuído manualmente.
- **Parte de interface**: formada pelo componente DCR de resolução de colisão e as conexões AUI e par trançado. Existe também a interface interna com o PC, que é a conexão nos slots do micro.

A placa possui uma FIFO de 128 bytes para a recepção e uma FIFO de 64 bytes para a transmissão.

O componente DCR é composto por um chip que implementa o algoritmo de busca em árvore binária para arbitrar de maneira determinista a resolução de conflitos na rede. De acordo com o contexto, o DCR pode estar ativado ou desativado. No último caso a placa comporta-se como uma placa de rede Ethernet clássica. A ativação se faz por meio de micro-chaves.

Existe um arquivo que acompanha o software chamado "defaults.mac"onde se realizam algumas configurações, tais como, o nível de interrupção utilizado para a recepção, o nível de requisição de DMA, o endereço de base da zona I/O, definir ou não modo determinista, e outros. Este arquivo estabelece valores defaults, mas alguns destes valores podem ser estabelecidos por micro-chaves, valor que prevalece.

Partindo do software que acompanha a placa Dethernet 104("Evaluation Dethernet/104 V. 0.2 Copyright 1993 - APTOR-ITMI") foram disponibilizadas as rotinas de transmissão e recepção de dados através das tarefas servidoras de rede, as quais utilizam os serviços fornecidos pela camada MAC implementados através do driver da interface de rede.

5.7 Descomposição da camada de enlace em módulos

Na figura 5.11 se mostra a estrutura proposta para o suporte de software da camada de enlace especificada no presente trabalho. Nesta estrutura se associa um módulo para cada sub-camada, os quais embutem tarefas associadas aos diferentes serviços fornecidos pelas camadas. Ditas tarefas apresentam portos de entrada e portos de saida para estabelecer a conexão lógica com outras tarefas pertencentes ao mesmo módulo ou para comunicar-se com outro módulo, como é o caso da conexão que se estabelece entre o módulo da camada MAC e o módulo da camada LLC. Como primitivas de comunicação são utilizadas, para a recepção, do tipo de primitivas de recepção bloqueante, e para o envio, do tipo de envio não bloqueante.

No módulo da camada LLC, as tarefas representam os possíveis estados em que se pode encontrar a entidade LLC, eles são:

- tarefa "wait": realiza a triagem da camada LLC encaminhando para a função solicitada, ou seja, a requisição de serviço da entidade LLC, ou o pedido de indicação à camada de aplicação da recepção de uma mensagem remota ou a recepção do estado da transmissão remota efetuada. Possui três portos de entrada para a recepção das

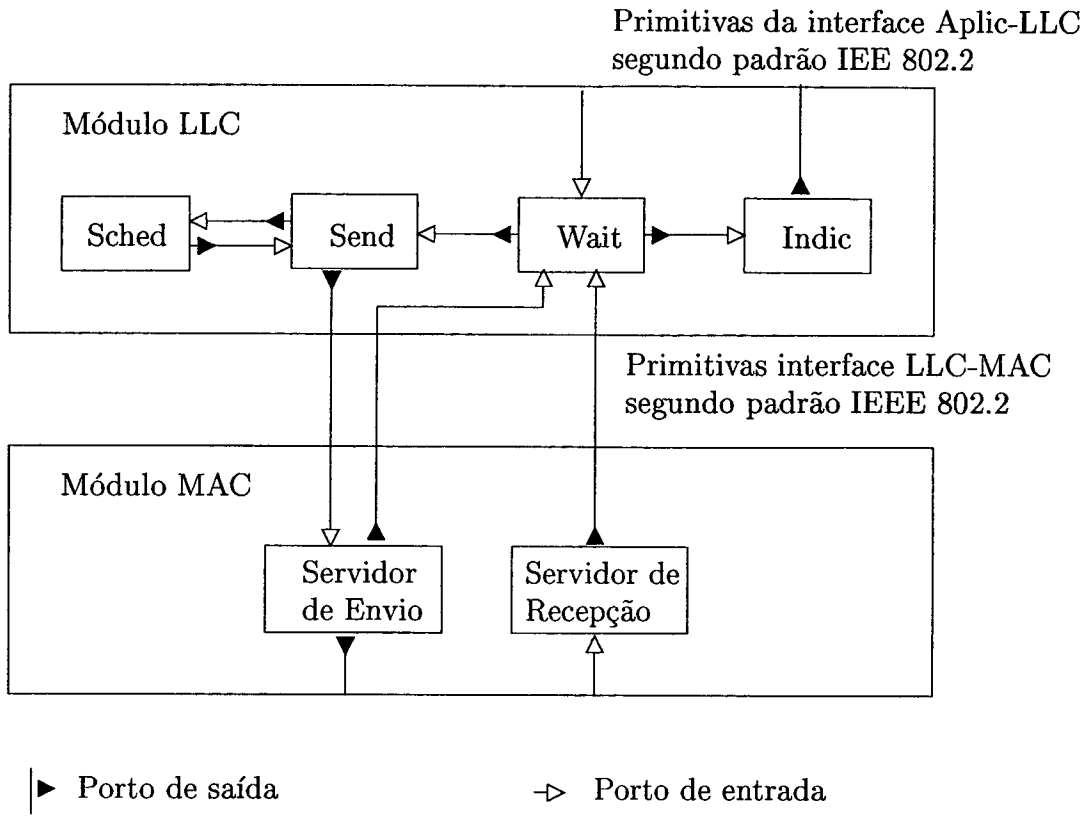

Figura 5.11: Descomposição em módulos da camada de enlace

mensagens anteriormente citadas, estabelecendo conexões lógicas com a camada de aplicação e com a camada MAC. Também possui dois portos de saida para estabelecer a comunicação com as tarefas "send"e "ind", dependendo da função solicitada, ou seja se se trata da transmissão remota de uma mensagem ou da indicação de chegada de uma mensagem remota.

- tarefa "send": nesta tarefa se chamam à três funções, uma primeira que realiza a função de fragmentação no caso de que o tamanho da mensagem que se quer transmitir a uma estação remota seja maior que o tamanho máximo reservado para o envio de dados no quadro definido pelo protocolo MAC, no caso, descontando os bytes reservados para informação característica do protocolo, restariam 1492 bytes para ser utilizados para a transmissão de dados. Os N pacotes que formam a mensagem devem ser sequênciados (o serviço sem conexão e sem reconhecimento não suporta no protocolo de enlace quadros numerados).

  Outra função chamada nesta tarefa é a encarregada de calcular o deadline do pacote, ou seja, se a mensagem deve ser fragmentada, deve ser calculado um novo deadline a partir da distribuição da folga da mensagem original entre os N pacotes em que

foi fragmentada a mensagem. Esta técnica se detalha na seção 5.2.3.2.

Uma vez obtidos os N pacotes com seus respectivos deadlines está-se em condições de chamar a função que formata os pacotes de acordo ao formato padrão de uma LLC PDU, a proposta de formato de LLC PDU se mostra na figura 5.5. Na figura do formato se observa que são adicionados os bytes correspondentes à informação de sequênciamento e o deadline de pacote.

A tarefa "send" possui um porto de entrada que recebe a mensagem original da tarefa "wait", um porto de saida para enviar a mensagem fragmentada e formatada como uma LLC PDU para a camada MAC à tarefa "servidor de envio", e finalmente um porto de saida e um porto de entrada para para estabelecer a comunicação com a tarefa "sched" que se ocupa da ordenação da fila de transmissão dos pacotes.

- tarefa "sched": esta tarefa realiza a ordenação na fila de transmissão, é ativada cada vez que se adiciona um pacote na fila de transmissão. O cálculo das prioridades dos pacotes na fila de transmissão se realiza utilizando a técnica proposta por Spuri para diminuir o custo de quantização produto do limitado número de niveis de prioridades existente, esta técnica se detalha na seção 3.6. Esta tarefa possui um porto de entrada e um porto de saida para comunicar-se com a tarefa "send".

- tarefa "indic": esta tarefa chama a função que realiza a blocagem dos pacotes recebidos através de seu porto de entrada da tarefa "servidor de recepção", obtendo assim a mensagem original a qual se envia através de seu porto de saida para a camada de aplicação.

No módulo da camada MAC se encontram embutidas dois tarefas, as tarefas servidoras de rede. É importante destacar que as primitivas de interface da camada LLC com a camada MAC adotam o padrão IEEE 802.2.

Para estabelecer uma comunicação remota utilizando um servico sem conexão e sem reconhecimento, a tarefa "send" envia uma mensagem através de primitivas não bloqueantes para um porto de saida, este porto mantém uma estrutura de ligação que contém o endereço local do porto de entrada da tarefa "servidor de envio" e o endereço de sistema do porto de entrada remoto. Uma vez que a tarefa "servidor de envio" recebe a mensagem, formata o descritor de mensagem remota, quem através da interface de rede, envia a mensagem a estação remota. Quando uma tarefa "servidor de recepção" recebe um pacote, executa a operação de envio para a tarefa "wait", quem encaminha a dita mensagem para realizar a indicação de chegada de mensagem para a camada de aplicação.

- tarefa "servidor de envio": quando um porto de saida é conectado a um porto de entrada, sendo que este porto de entrada é remoto, o endereço local mantido na estrutura de ligação corresponde ao ponteiro para o descritor do porto de entrada da tarefa "servidor de envio". A tarefa "servidor de envio"recebe a mensagem no porto de entrada, colocando ela num buffer, para logo depois realizar a formatação do quadro de acordo ao protocolo MAC utilizado, para finalmente enviar através de seu porto de saida o quadro formatado utilizando o serviço "enviar pacote", disponível na interface do driver de rede.

- tarefa "servidor de recepção": esta tarefa é ativada por interrupção, ou seja, sempre que chega um quadro ou pacote na estação se gera uma interrupção com o objetivo de viabilizar a recepção o mais brevemente possível, liberando desta forma a recepção para outros pacotes ou evitar a perda de pacotes devido a que o buffer de recepção esté ocupado. A interrupção deve ser habilitada no tempo de configuração pela interface de rede, assim quando chega um pacote, a tarefa "servidor de recepção"é acordada, realizando a operação de envio através do porto de saida para a tarefa "wait". A rotina de interrupção para a placa de rede Dethernet 104 do fabricante APTOR/ITMI está em assembler, e o vetor de interrupção que pode ser utilizado para a recepção é ajustado no arquivo "defaults.mac", o qual acompanha o driver da interface de rede, e sua leitura se faz no tempo de configuração da interface de rede.

5.8 Análise de escalonabilidade

Para a análise de escalonabilidade utilizaremos a técnica abordada por Spuri em [Spu96]. Apresentaremos o modelo de escalonamento utilizado, ou seja, o modelo de mensagens, o modelo do recurso e a política de escalonamento utilizada. Como política de escalonamento utiliza-se o EDF("Earliest Deadline First"), isto é, a mensagem que possui o deadline mais próximo do tempo corrente atribui-se maior prioridade.

5.8.1 Modelo de mensagens

A continuação se apresentam as características do conjunto de mensagens conhecidas a priori que definem a carga do sistema analisado:

- está formado por um conjunto de mensagens periódicas, cujos tempos de chegada na fila de transmissão estão separados por um intervalo mínimo T, chamado de período.

- cada mensagens m_i possui um tempo máximo e limitado de transmissão, chamado de C_i. Este tempo depende do comprimento da mensagem e da taxa de transmissão entre os nós.
- cada mensagem possui um deadline D_i, que é o limite máximo para que se complete a transmissão entre o nó origem e o nó destino.
- as mensagens são conhecidas a priori, conformando uma carga estática, o qual permite a realização de um escalonamento com garantia de projeto.
- se permitem deadlines arbitrários, ou seja, as mensagens podem apresentar deadlines menores, iguais ou maiores à seus respectivos períodos.
- as mensagens são fragmentadas em pacotes ou frames , o tamanho máximo estará em função do protocolo MAC implementado(CSMA-DCR).
- as mensagens podem apresentar jitter.

5.8.2 Modelo de comunicação

Modelando o recurso, expressamos como suas características temporais influênciam no atraso fim-a-fim da transmissão de mensagens.

Um número limitado de estações estão interconectadas através de um barramento, cada estação acede o canal por meio de um adaptador de comunicação, quem implementa o protocolo de comunicação para a camada MAC, neste trabalho utiliza-se o CSMA-DCR.

O acesso ao barramento é arbitrado como um protocolo CSMA-CD até que se produz uma colisão. Quando ocorre uma colisão realiza-se uma busca na árvore binária balanceada, e a ordem das transmissões acima do canal compartilhado se darão em função do índice alocado pelas estações envolvidas na colisão.

Cada estação tem o direito a transmitir um tamanho máximo de pacote ou frame. As mensagens antes de ser entregue ao nível MAC podem ser fragmentadas em pacotes, que em função do protocolo de comunicação, define-se um tamanho máximo de pacote. Os pacotes herdam as restrições temporais da mensagem original.

O adaptador de comunicação e o processador de cada estação compartilham a fila de pacotes a ser transmitidos, ordenados por prioridade, a qual depende do deadline da mensagem original. O processador coloca os pacotes na fila ordenada e são removidos pelo adaptador imediatamente depois que a estação adquire o direito a transmitir. Uma vez iniciada a transmissão de um pacote esta não pode ser interrompida. A nível local

esta permitida a preempção da mensagem que esta sendo transmitida, por outra de maior prioridade, mas somente nos limites dos pacotes.

Quando o pacote chega ao adaptador de comunicação do nó destino, é colocado num "buffer", gerando uma interrupção para avisar ao processador, quem remove, processa ou libera o pacote.

5.8.3 Análise para o protocolo CSMA-DCR

No modelo de escalonamento, as ocorrências de mensagens são tratadas como ocorrências de tarefas.

Esta abordagem baseia-se no cálculo do pior caso de tempo de resposta de uma mensagem i, ou o tempo máximo de transmissão de uma mensagem i. Quando uma mensagem excede o tamanho máximo de pacote, ela é fragmentada, então calculando o tempo de resposta do último pacote, estamos calculando o tempo de resposta da mensagem, cujo deadline deve ser maior ou igual que o pior caso de resposta de uma mensagem para ser garantida:

$$r_i = max_{a \in [-J_i, L-J_i-C_i-B_i)}\{r_i(a)\}$$

onde a representa o tempo de chegada da mensagem i. O pior tempo de resposta relativo à a é:

$$r_i(a) = max\{J_i + C_i + B_i, L_i(a) - a\}$$

Da expressão anterior se deduz que para calcular o pior tempo de resposta deve-se calcular o "busy period"$(L_i(a))$, então temos que:

$$\begin{cases} L_i^{(0)}(a) = \sum_{\substack{j \neq i \\ D_j \leq a + D_i + Jj}} C_j + I_{\{s_i(a)=0\}} C_i, \\ L_i^{(m+1)}(a) = W_i(a, L_i^{(m)}(a)) + B_{i(a+D_i)} \end{cases}$$

A equação acima converge em um finito número de passos se a utilização do recurso dos i níveis mais altos de prioridade é menor que a unidade, ou seja:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1$$

É importante destacar que deve ser considerando o máximo custo de quantização x em termos de utilização, produto do limitado número de níveis de prioridade existente, igual a 2^n. Para isto se considera uma carga máxima(k_{max}), e se o custo supera a unidade se

diz que não é possível garantir a escalonabilidade do conjunto de mensagens. O máximo custo de quantização se calcula da seguinte forma:

$$x = \frac{1}{\left\lfloor \frac{2^n}{\lfloor \log_2 k_{max} \rfloor + \frac{k_{max}}{2^{\lfloor \log_2 k_{max} \rfloor}}} \right\rfloor}$$

Este custo deve ser considerado como uma penalização na utilização do sistema, portanto, a condição para a convergência da equação de cálculo do "busy period"deve ser modificada considerando dito custo de quantização x, obtendo-se:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq 1 - x$$

Para calcular o "busy period"é necessário calcular a carga de trabalho acumulada, temos então que:

$$W_i(a,t) = \sum_{\substack{j \neq i \\ D_j \leq a + D_i}} min\left\{ \left\lceil \frac{t}{T_j} \right\rceil, 1 + \left\lfloor \frac{a + D_i - D_j}{T_j} \right\rfloor \right\} C_j + \delta_i(a,t) C_i,$$

onde

$$\delta_i(a,t) = \begin{cases} min\left\{ \left\lceil \frac{t - s_i(a)}{T_i} \right\rceil, \; 1 + \left\lfloor \frac{a}{T_i} \right\rfloor \right\} & se \; t > s_i(a), \\ 0 & outros \end{cases}$$

Até aqui, nesta seção, foi apresentada de forma sucinta a proposta de Spuri abordada em [Spu96] para a análise de escalonabilidade, mas restringida ao modelo de mensagens especificada anteriormente. A continuação, além de especificar alguns termos que aparecem nas equações acima, como C_i e B_i, se adicionam outros termos característicos do protocolo MAC utilizado, que também devem ser considerados e especificados.

5.8.3.1 Modelagem do protocolo MAC

Existe um tempo, característico do protocolo CSMA-DCR, onde a estação k deve esperar para adquirir o direito a transmitir a mensagem m_i que possui na cabeça da fila de emissão. Identificando este tempo como um tempo onde a estação k está impossibilitada de aceder o meio porque ele esta sendo utilizado pelas transmissões de outras estações, ou se está produzindo a resolução da árvore binaria(slot-times vazios ou colisão). Como se

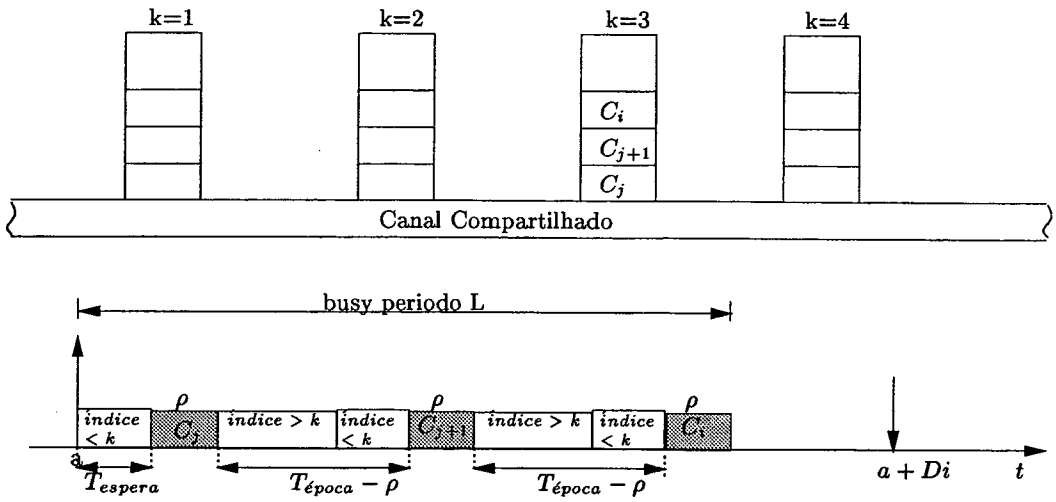

Figura 5.12: Modelagem do algoritmo MAC

observa na figura 5.12, este tempo afeta a transmissão de cada pacote da fila, portanto no caso de querer transmitir a mensagem m_i, deve-se considerar que isto acontece para todos os pacotes que antecedem o pacote i. Então se definimos $\theta(a,t)$ como uma tarefa de maior prioridade que a transmissão da mensagem m_i, com deadline igual ao tempo $T_{época}$ e cujo tempo de computação é igual a $T_{época} - \rho$ temos que:

$$\theta_i(a,t) = \begin{cases} min\left\{\left\lceil \frac{t}{T_{época}} \right\rceil, 1 + \left\lfloor \frac{a+D_i-T_{época}}{T_{época}} \right\rfloor\right\}(T_{época} - \rho) & se\ a + Di > T_{época}, \\ 0 & outros \end{cases}$$

5.8.3.2 Modelagem do bloqueio

Para calcular o pior caso de tempo de bloqueio por inversão de prioridades, ou seja, o tempo que um pacote de maior prioridade espera pela transmissão de um pacote menos prioritário temos que considerar a situação seguinte, o pacote i de maior prioridade, chega na fila de transmissão no instante imediato posterior ao começo da "época"da qual ele não faz parte, obrigado a esperar o termino desta "época", na que sim participa o pacote de menor prioridade, temos então que no pior caso é:

$$B_1 = T_{época}$$

Existe outro bloqueio produto do modo de operação do protocolo MAC como época bloqueada ou fechada, ou seja, quando um pacote chega na fila de transmissão no instante imediato posterior ao começo da "época" da qual ele não faz parte, obrigado então a esperar o termino desta "época". Este bloqueio não se produz junto com o bloqueio por inversão de prioridades, eles são excludentes, mas os dois no pior caso são iguais ao tempo de um

"época".

$$B_2 = T_{época}$$

Existe finalmente um terceiro bloqueio, associado ao tempo que deve esperar uma estação para transmitir seu pacote produto da resolução da árvore, chamado de T_{espera}, este bloqueio se adiciona ao bloqueio por inversão de prioridades ou ao bloqueio por época fechada, e seu valor está em função do índice da estação, isto é:

$$B_3 = T_{espera}(k) = \varphi(k)s + k\rho$$

onde $\varphi(k)$ representa o número de ramos da árvore binária percorridos por uma mensagem proveniente de um nó com índice k, e s o tempo de slot-time.

Concluindo, o bloqueio não afeta o trabalho cumulativo, entretanto incrementa o tamanho do "busy period"e pode ser expressado como:

$B_i = B_1 + T_{espera}(k)$ ou $B_i = B_2 + T_{espera}(k)$

onde tanto B_1 como B_2 são iguais a $T_{época}$, ou seja,

$$B = T_{época} + T_{espera}(k)$$

5.8.3.3 Sobrecarga produto do encapsulamento

O tempo de transmissão de uma mensagem C_i^* é igual à suma dos tempos de transmissão dos pacotes desprendidos da fragmentação. Mas como se observa na figura 5.5, existe uma sobrecarga produto do encapsulamento dos pacotes que se produz no nível MAC. Refere-se ao tempo de transmissão dos bits que não são dados, ou seja, os bits que carregam outro tipo de informação exclusiva do protocolo, como endereço fonte, endereço destino, CRC e outros. Este tempo representa uma sobrecarga que afeta o tempo de transmissão da mensagem.

Se se define ρ como o tempo de transmissão de um quadro máximo e considerando que este tempo inclui o tempo de transmissão dos bits de dados do quadro, especificado como C_{dados}, mais o tempo de transmissão da sobrecarga produto do encapsulamento, especificado como C_{enc}, temos que:

$$\rho = C_{dados} + C_{enc}$$

Considerando que o número de pacotes, especificado como N_i, que conformam uma mensagem se calcula como:

$$N_i = \frac{tempo\ transmissão\ mensagem}{tempo\ transmissão\ bits\ de\ dados\ do\ pacote} = \frac{C_i}{C_{dados}}$$

temos então, que o tempo de transmissão de uma mensagem considerando o encapsulamento é:

$$C_i^* = \rho\ N_i$$

Neste tempo encontra-se embutido o tempo de encapsulamento da mensagem, o qual pode ser facilmente extraído calculando $N_i\ C_{enc}$.

5.8.3.4 Construindo o teste de escalonabilidade

Concluindo, para que uma mensagem m_i de uma estação com índice k seja escalonável seu deadline deve ser menor ou igual ao pior tempo de resposta da mensagem, ou seja:

$$d_i \leq r_i$$

onde

$$r_{i,k} = max\ r_{i,k}(a)$$

$$r_{i,k}(a) = max_{a \in [-J_i, L-J_i-C_i^*-B_i)}\{r_{i,k}(a)\}$$

e

$$r_{i,k}(a) = max\ C_i + J_i + B_i, L_i(a) - a$$

Considerando então:
- que $B_i = T_{época}$
- que $C_i^* = \rho\ N_i$
- o $\theta_i(a,t)$

temos que o "busy period"é:

$$\begin{cases} L_{i,k}^{(0)}(a) = \sum_{j \neq i} N_{j,k}\rho + I_{\{s_{i,k}(a)=0\}} N_{i,k}\rho + \theta_{i,k}(a,t), \\ L_{i,k}^{(m+1)}(a) = \underset{D_{j,k} \leq a + D_{i,k} + J_{j,k}}{W_{i,k}(a, L_{i,k}^{(m)}(a))} + T_{época} + T_{espera} \end{cases}$$

onde o trabalho cumulativo pode ser calculado como:

$$W_{i,k}(a,t) = \sum_{\substack{j \neq i \\ D_{j,k} \leq a + D_{i,k}}} min\left\{\left\lceil \frac{t}{T_{j,k}} \right\rceil, 1 + \left\lfloor \frac{a + D_{i,k} - D_{j,k}}{T_{j,k}} \right\rfloor\right\} N_{j,k}\rho + \delta_{i,k}(a,t) N_{i,k}\rho + \theta_{i,k}(a,t),$$

$\theta_{i,k}(a,t)$ se calcula como:

$$\theta_{i,k}(a,t) = \begin{cases} min\left\{\left\lceil \frac{t}{T_{época}} \right\rceil, 1 + \left\lfloor \frac{a + D_{i,k} - T_{época}}{T_{época}} \right\rfloor\right\} (T_{época} - \rho) & se\ a + D_{i,k} > T_{época} \\ 0 & outros \end{cases}$$

e $\delta_{i,k}(a,t)$ se calcula como:

$$\delta_{i,k}(a,t) = \begin{cases} min\left\{ \left\lceil \frac{t - s_{i,k}(a)}{T_{i,k}} \right\rceil,\ 1 + \left\lfloor \frac{a}{T_{i,k}} \right\rfloor \right\} & se\ t > s_{i,k}(a), \\ 0 & outros \end{cases}$$

5.8.4 Análise para mensagens com relação de precedência

Dentro dos tipos de quadros suportados no tipo de serviço *tipo 1* temos o UI, TEST e XID. O tipo de quadro UI somente se apresenta como comando, no entanto, os quadros TEST e XID apresentam-se como quadros tipo comandos e respostas, gerando uma relação de precedência para este tipo de mensagens. Por isto, apesar de utilizar um serviço sem conexão e sem reconhecimento, ao basear-se no padrão IEEE 802.3, surge a necessidade de modelar mensagens com relação de precedência para incluir este tipo de quadros. Estes comandos são opcionais no momento da implementação, mas é obrigatoria a resposta à recepção de um deles.

As mensagens tipo respostas terão que disputar os recursos com as demais mensagens do nó, ou seja desde o ponto de vista do nó que a transmite ela é uma mensagem independente. No entanto desde uma visão mais global, esta mensagem depende do comando que está esperando uma resposta, e a resposta somente vai ser gerada no caso da recepção do comando, gerando uma relação de precedência entre elas. Para garantir que o nó destinatário do comando envie uma resposta tão breve quanto possível, uma possibilidade é associar á mensagem resposta uma alta prioridade.

Para analisar o pior tempo de resposta de uma mensagem tipo resposta imaginemos uma situação de instante crítico, onde a mensagem tipo resposta pode sofrer um atraso na sua liberação, dependendo do tempo de transmissão do comando, isto é, seu "relea-

se jitter"será igual ao pior caso de tempo de envio da mensagem tipo comando. Para considerar dito fato na análise de escalonabilidade, na expressão de cálculo do trabalho cumulativo deve-se considerar que:

$$J_{resposta} = r_{comando}$$

adicionando o termo $J_{resposta}$ para as mensagens tipo respostas e anulando ele para as outras.

5.8.5 Exemplo de um conjunto de mensagens

Consideremos o seguinte exemplo, um conjunto de mensagens com deadlines menores, iguais e maiores que seus respectivos períodos. A mensagem m_4 é uma mensagem tipo resposta da mensagem m_3, por tanto apresenta um "release jitter"igual ao pior tempo de resposta da mensagem m_3. Para a análise consideremos o pior padrão de chegada, isto é, a primeira instância de todas as mensagens são liberadas no tempo *t=0*, e as outras instancias são liberadas de acordo com os períodos das mensagens.

Considerando uma rede com 64 estações, onde os parâmetros da interface que implementa o protocolo MAC são, um tempo de *slot-time* de $60\mu seg$, o tamanho de quadro máximo é de 1418 bytes, mas o número de bytes para enviar dados é de 1485, e 29 bytes de encapsulamento, então temos:

- $s = 60\mu seg$
- $q = Q = 64$
- o índice da estação se corresponde com o índice mais alto, ou seja a estação menos prioritária na rede, isto é, $k = 63$
- o número de bytes de dados no pacote de tamanho máximo é de 1485, e o número de bytes de encapsulamento é de 29, portanto para calcular o tempo C_{dados}, e o tempo C_{enc} para uma velocidade de transmissão de 10Mbis/seg temos:

$C_{dados} = 1485(0.8\mu seg) = 1188\mu seg$ e $C_{enc} = 29(0.8\mu seg) = 24\mu seg$

- $\rho = C_{dados} + C_{enc} = 1212\mu seg$
- $T_{época} = \varphi(q-1)s + Q\rho = 81.348\mu seg$
- $T_{espera}(k) = \varphi(k-1)s + k\rho = \log_2 64 + (63-1) - \sigma(63-1) = 80.136\mu seg$
- $B_i = T_{época} + T_{espera}(k) = 161.484\mu seg$

O conjunto de mensagens analisado se mostra na tabela 5.6, onde todos os tempos estão expressados em microsegundos.

A tabela 5.6 representa um de vários exemplos realizados utilizando o software MATLAB. Como o teste de escalonabilidade adotado para a análise suporta deadlines arbi-

i	D_i	T_i	C_i	J_i	r_i
1	260.000	270.000	500	0	162.696
2	350.000	350.000	800	0	324.180
3	420.000	420.000	923	0	405.528
4	500.000	400.000	1004	r_3	486.876

Tabela 5.6: Conjunto de mensagens

trários, no conjunto de mensagens se representam todas as situações, assim como também foi considerada que a mensagem m_4 é uma mensagem tipo resposta, ou seja uma mensagem com relação de precedência, o qual eqüivale a ter que considerar seu *jitter* igual ao tempo de resposta da mensagem comando que a gerou.

É importante destacar que os cálculos foram realizados considerando que este conjunto de mensagens se encontram na estação com o índice mais alto, ou seja está-se representando o pior caso dentro dos índices possíveis.

Do conjunto de mensagens analisado pode-se concluir que o pior tempo de resposta para cada mensagem m_i é menor que seu respectivo deadline, sendo possível então escalonar dito conjunto de mensagens.

5.9 Considerações sobre a proposta

A proposta para sistemas tempo real apresentada por Arvind em [ARS91], possui como principal contribuição, o esquema proposto para a escolha da combinação de protocolos MAC e LLC a partir da sua classificação pelo tipo de abordagem utilizada para a garantia, mas o critério que usa para a escolha de serviços que garantem a entrega das mensagens também é uma limitação desta proposta, pois as mensagens que exigem garantia estão associadas somente a um serviço orientado a conexão, enquanto as mensagens "best effort" estão associadas a um serviço sem conexão. Outra limitação desta proposta é que, embora marca a importância da correta identificação e especificação dos requisitos de comunicação, não explicitam a forma de como estes requisitos devem ser representados.

Como o objetivo é construir um teste de escalonabilidade modelando recursos concretos, se busca simplificar a análise, para o qual optou-se pelo serviço mais simples, ou seja, serviço sem conexão e sem reconhecimento, de forma de não inviabilizar a análise de escalonabilidade, para isto assume-se que dito serviço não envolve perdas de mensagens. Esta premissa pode-se conseguir na prática usando técnicas de replicação a nível MAC e físico.

Outro aspecto considerado nesta proposta de camada de enlace é a necessidade de um

escalonador de pacotes dirigido a evento. Detalhando a implementação de um suporte que permita o escalonamento dos pacotes em função das prioridades. É importante salientar a influência do relógio da estação, no sentido de como sua granularidade limita as restrições temporais, para limitar este custo resulta interessante utilizar sistemas operativos orientados a aplicações tempo real, como exemplo temos a versão do LINUX que permite uma resolução do timer na ordem dos microsegundos.

Em relação ao uso do EDF e dos testes propostos por Spuri, os vários exemplos realizados mostram a viabilidade do uso desse teste para a análise de escalonabilidade, o qual, por ser um teste *off-line* não representa custo de implementação, sendo o maior problema a representação do deadline com um número limitado de níveis de prioridade, mas isto é considerado neste trabalho, adotando a técnica proposta por Spuri em [MNS96] para diminuir este custo de quantização.

Que se tenha conhecimento este é o único esforço usando o EDF como política de escalonamento para a comunicação de mensagens e utilizando para a análise de escalonabilidade a técnica abordada por Spuri em [Spu96].

5.10 Conclusões

Neste capítulo apresenta-se uma proposta de camada LLC para comunicação tempo-real. A proposta baseia-se no padrão ANSI/IEEE 802.2(ISO 8802-2) [fSEC94], sendo esta uma contribuição deste capítulo já que se especifica uma camada LLC totalmente padronizada. Dentro do padrão IEEE 802.3 especificou-se o tipo de serviço sem conexão e sem reconhecimento, ou seja uma operação tipo 1.

A vantagem de utilizar um padrão para a especificação da camada LLC é que a camada de aplicação independe da forma como foi desenvolvida a camada LLC, tendo somente que preocupar-se com a interface especifica para o tipo de serviço solicitado.

A partir daqui se apresenta a análise de escalonabilidade utilizando uma política EDF com a técnica apresentada em [Spu96] para o protocolo determinista CSMA-DCR.

A contribuição principal deste capítulo é o detalhe do processo de construção do teste de escalonabilidade para um modelo de mensagens que admite, deadlines arbitrários, "release jitter"e compartilhamento de recursos. Embora a especificação da camada seja para um serviço sem conexão e sem reconhecimento, ao mesmo tempo surge a necessidade de considerar mensagens com relação de precedência, devido a que o padrão tomado como referência possui quadros tipos comando e resposta.

Outro aspecto importante abordado neste capítulo, e considerado na hora do teste de escalonabilidade, é o custo de quantização produto do número limitado de níveis de

prioridade, se explica como este custo produz uma depreciação na utilização do meio de comunicação, que no caso de o custo ser maior que a unidade, inviabiliza a escalonabilidade do conjunto de mensagens.

Capítulo 6

Conclusões e Perspectivas

6.1 Conclusões

Este trabalho apresenta as especificações de uma camada de enlace para um sistema de comunicação tempo real, que utiliza um serviço sem conexão e sem reconhecimento, utilizando o protocolo determinista CSMA-DCR.

Se estabelecem os conceitos básicos dos sistemas tempo real, os requerimentos que os caracterizam assim como os diferentes critérios que podem ser adotados para sua classificação. Partindo destes conceitos pode-se dizer que o sistema abordado no presente trabalho é um STR crítico com carga estática.

Trata-se de uma rede local de vários computadores com uma arquitetura distribuída, tendo que considerar com especial atenção a problemática do sistema de comunicação, devido à grande influência dele no possível cumprimento dos requerimentos temporais das mensagens. Definindo um sistema de comunicação tempo real como aquele sistema que atende os requisitos temporais de comunicação das mensagens.

A arquitetura para sistemas tempo real adotada é apresenta em [ARS91], descrevendo as funções de cada camada, assim como os serviços e protocolos que implementam eles. Mas como a proposta de [ARS91] associa o serviço sem conexão e sem reconhecimento as abordagens melhor esforço, e o que se busca é obter um teste de escalonabilidade, optou-se começar pelo serviço mais simples, o serviço sem conexão e sem reconhecimento mas com garantia de entrega das mensagens dentro de seus deadlines, assumindo que não existem perdas de mensagens.

A garantia do cumprimento das restrições temporais sustenta-se na hipótese que se faz de considerar, o sistema de comunicação tempo real submetido a uma carga estática de mensagens, o qual permite o conhecimento a priori, em tempo de projeto, das características do tráfego na rede, permitindo calcular o tempo máximo de latência das

mensagens em função das características físicas da rede e o protocolo que controla o acesso ao recurso compartilhado, o suporte de comunicação.

Uma questão chave na problemática dos sistemas de comunicação tempo real é o escalonamento das mensagens nos recursos de comunicação do sistema. Para abordar este aspecto, apresentam-se diferentes propostas que partem de hipóteses e política de escalonamento diferentes. Resulta mais fácil encontrar na literatura propostas que utilizam políticas de escalonamento com atribuição de prioridades fixas para o envio das mensagens. Neste trabalho se apresentam dois algoritmos que trabalham com prioridades fixas ([TBW94], [KS94]) e um com prioridades dinâmicas ([Spu96]), adotando este último como algoritmo de escalonamento no modelo de escalonamento do sistema de comunicação tempo real.

A seleção da proposta de [Spu96] como algoritmo de escalonamento utilizado, baseia-se na característica de apresentar uma melhor utilização do sistema, comparando com os algoritmos que utilizam prioridades fixas na atribuição das prioridades. Além é uma proposta completa, no sentido de considerar como em [TBW94], um modelo computacional que admite tarefas periódicas com deadlines arbitrários, estendendo o modelo a tarefas esporadicamente periódicas (tarefas tipo "bursty") com "release jitter", e compartilhando recursos.

Estamos então, frente a um sistema de comunicação tempo real que utiliza um escalonador dinâmico(determina a escala de transmissão das mensagens em tempo de execução) aplicado a uma carga estática, onde a análise de escalonabilidade se faz considerando o instante crítico de envio das mensagens, o qual deriva em um cálculo considerando o pior caso(pior tempo de resposta) na fase de projeto.

O calculo do pior tempo de resposta é de grande importância nos sistemas tempo real distribuídos, porque as aplicações distribuídas são caracterizadas pela relação de precedência entre suas mensagens, então, se se trata de fazer uma alocação local sobre um suporte de comunicação distribuído, assume-se que o "release jitter"de uma mensagem pode estar demorado pela espera de chegada da mensagem predecessora, calculando-se então este "release jitter"partindo do pior tempo de resposta da mensagem predecessora proveniente de outra estação.

A especificação da camada de enlace conforma-se de vários elementos, primeiro, da análise das características temporais do protocolo MAC que arbitra o acesso ao recurso compartilhado, segundo, da proposta da camada LLC em função do tipo de serviço que se solicita e por ultimo, da definição do algoritmo de escalonamento utilizado na análise de escalonabilidade.

O protocolo adotado para arbitrar o acesso ao canal de comunicação compartilhado, o

CSMA-DCR, apresenta benefícios evidentes se comparar-mos com outros protocolos que podem ser utilizados em aplicações tempo real, como é que até a ocorrência da colisão funciona como um CSMA/CD, passando a utilizar largura de banda somente depois da colisão, além de apresentar índices aceitáveis de desempenho, no sentido do tempo máximo de transmissão de um quadro, como da porcentagem da banda passante que é utilizada para a transmissão de informação útil. Outra característica é a robustez, no sentido que o protocolo é totalmente repartido, não sendo necessário a existência de alguma ficha susceptível a ser perdida, não sendo necessário nenhum procedimento específico para a inserção o saída de alguma estação na rede. Se apresenta como um protocolo auto adaptativo no sentido que se adapta automaticamente à carga da rede.

A proposta da camada LLC se faz sobre a base do padrão ANSI/IEEE 802.2(ISO 8802-2). Dos tipos de serviço fornecidos pela camada LLC, utiliza-se o tipo 1(sem conexão e sem reconhecimento), detalhando as primitivas que permitem seu acesso, os diagramas de estado dos componentes que conformam a entidade LLC, assim como os procedimentos que resultam do conjunto de comandos e respostas enviados. Resulta interessante destacar que foram considerados a existência de quadros com relação de precedência, expressando-se como comandos e respostas. Sendo obrigatório a resposta à chegada de um comando, e quedando a critério do projetista a utilização dos comandos.

Partindo do sistema acima especificado, define-se o modelo de escalonamento utilizado, estabelecendo o modelo das mensagens, o modelo do recurso compartilhado e a política de escalonamento para realizar a análise de escalonabilidade das mensagens, obtendo a expressão da condição de escalonabilidade.

Outra questão considerada no presente trabalho é como o uso de um limitado número de níveis de prioridades produz inversão de prioridades, expressando como dito efeito se reflexa como uma redução na utilização do canal, propondo uma técnica para diminuir este efeito.

A principal contribuição do presente trabalho é a especificação da camada de enlace combinando dois elementos que apresentam boas características, como são, o protocolo CSMA-DCR e a utilização de um algoritmo de escalonamento com prioridades dinâmicas. Além da contribuição que representa no sentido de obter um teste de escalonabilidade que modela recursos concretos e utiliza o EDF como política de atribuição de prioridades, a qual tem sido menos abordada na literatura se comparamos com a política de prioridades fixas.

6.2 Perspectivas

Como perspectivas de trabalho podem ser propostas várias linhas como:

- completar a especificação desta camada de enlace, mas adicionando as hipóteses que não foram consideradas neste trabalho, como é considerar a existência de tarefas aperiódicas.
- tomando o padrão ANSI/IEEE 802.2(ISO 8802-2) como referência, implementar a camada de enlace para serviços sem conexão e com reconhecimento.
- tomando o padrão ANSI/IEEE 802.2(ISO 8802-2) como referência, especificar a camada de enlace para serviços orientado a conexão.

Apêndice A

Estrutura LLC PDU

Na tabela seguinte se mostra o formato de uma LLC PDU(Protocol Data Unit)

DSAP	*SSAP*	*Controle*	*Information*
8 bits	8 bits	8 bits	N*8 bits

Tabela A.1: Formato de uma LLC PDU

DSAP : Campo que especifica o endereço do SAP destino, pode ser individual ou de grupo.

SSAP : Campo que especifica o endereço do SAP fonte.

Controle : Campo de controle.

Informação : Campo que carrega a informação.

A.1 Campos de endereçamento

Na tabela seguinte se observa o formato dos campos DSAP e SSAP.

DSAP								SSAP							
1/G	D	D	D	D	D	D	D	C/R	S	S	S	S	S	S	S

Tabela A.2: Formato dos campos DSAP e SSAP

1/G = 0	DSAP individual	C/R = 0	Comando
1/G = 1	DSAP de grupo	C/R = 1	Resposta

X0DDDDDD	Endereço DSAP	X1DDDDDD	Reservado
X0SSSSSS	Endereço SSAP	X1SSSSSS	Reservado

Considerações dos campos de endereçamento:

- Cada campo é formado por um byte.
- Os bytes dos campos de endereçamento contem 7 bits para os endereços propriamente ditos(DSAP e SSAP); e o bit menos significativo, no campo DSAP identifica se é um endereço individual ou de grupo, e no campo SSAP identifica se a LLC PDU é um comando ou resposta.
- Quando o campo DSAP contem todos seus bits em "1", define-se como um endereço global de destino.
- Quando os campos DSAP ou SSAP contem todos seus bits em "1", define-se como um endereço nulo, que identifica o LLC associado com o endereço MAC local.
- Os endereços X1000000 são designados para funções de gerenciamento da camada LLC local.
- Outros endereços onde o bit próximo ao menos significativo é "1", são reservados para definições da ISO.

A.2 Campo de Controle

O serviço Classe I pode-se aplicar à endereços individual, de grupo, global e nulos e aplicações que não requerem reconhecimentos dos dados nem controle de fluxo. O conjunto de comandos e respostas PDUs suportados por esta classe de serviço são os que se observam na tabela seguinte:

Comando	*Resposta*
UI	
XID	XID
TEST	TEST

Tabela A.3: Comandos e Respostas para operação tipo 1

A operação tipo 1 utiliza os quadros não numerados acima (UI, XID, TEST) com o formato da tabela A.3

1	2	3	4	5	6	7	8
1	1	M	M	P/F	M	M	M

Tabela A.4: Formatos dos quadros não numerados

M : bits identificadores de comando não numerado

P/F : poll bit, (P=1) solicitação de resposta imediata final bit, (F=1) indicador de resposta de solicitação imediata

O único parâmetro que existe na operação tipo 1 é o bit Poll/Final(P/F). E usado somente com os quadros XID e TEST.

A.2.1 Comandos e Respostas

1	2	3	4	5	6	7	8	bits
1	1	0	0	0	0	0	0	comando UI
1	1	1	1	P	1	0	1	comando XID
1	1	0	0	P	1	1	1	comando TEST

Tabela A.5: Comandos para operação tipo 1

1	2	3	4	5	6	7	8	bits
1	1	1	1	F	1	0	1	resposta XID
1	1	0	0	F	1	1	1	resposta TEST

Tabela A.6: Respostas para operação tipo 1

- UI

  O quadro *UI* só existe como comando. A recepção do comando *UI* não é reconhecida nem seqüenciada, portanto a informação pode ser perdida no caso de acontecer algum tipo de erro na transmissão, ou que o receptor este ocupado no momento de envio do comando *UI*. No comando *UI* o bit P deve ser "0".

- XID

  No comando *XID* o bit P pode ser "0" ou "1", mas a entidade LLC receptora deve enviar o mesmo valor que recebeu, no bit F. Este comando é opcional na implementação, mas é mandatorio a resposta à recepção de dele. O comando/resposta XID é utilizado para procedimentos de identificação de outras entidades LLC, assim como também para determinar os tipos de serviços suportados pelas entidades LLC remotas.

Possíveis usos do comando XID :

- O comando XID com endereços DSAP e SSAP nulos solicita a resposta de alguma estação.
- O comando XID com um endereço DSAP de grupo pode ser utilizado para determinar os membros de um grupo.
- O comando XID com um endereço DSAP global pose ser usado para identificar todas as estações ativas.
- Pode ser utilizado para determinar o tipos de serviços ou classes de LLC dos SAP especificados nos endereços. Para isto tem que se examinar o campo de informação do XID.

• TEST

O comando/resposta TEST pode ser utilizado para os testes de "loopback". Quando a informação recebida no campo de informação, pela entidade LLC emissora, é a mesma ao que ela enviou, pode-se concluir que o teste concluiu satisfatoriamente. Da mesma forma que com XID, este comando é opcional na implementação, mas é mandatorio a resposta à recepção de dele.

Referências Bibliográficas

[ABB+92] N. Audsley, A. Bhatlacharyya, A. Burns, G. Fohler, H. Kantz, H. Kopetz, J.A. McDermid, W. Schütz, and R. Zainlinger. *Timeliness - Summary and Conclusions - Specification and Design for Dependability.* ESPRIT BRA Proj 3092 vol.2, May 1992.

[ABNS98] P. Ancilotti, G. Buttazzo, M. D. Natale, and M. Spuri. *Design and Programming Tools for Time Critical Applications.* Real-Time Systems, vol 14, 3 pp 61-93, Kluwer Academic Publishers, 1998.

[ARS91] K. Arvind, K. Ramamritham, and J.A. Stankovic. *A Local Area Network Architecture for Communication in Distributed Real-Time Systems.* The Journal of Real-Time Systems, pp 115-147, Kluwer Academic Publishers, 1991.

[FFF95] K. V. O. Fonseca, J. M. Farines, and J. Fraga. *Um estudo comparativo de técnicas de análise de escalonabilidade em um sistema de comunicaçaõ tempo-real.* Anais do Simpósio Brasileiro de Redes de Comunicação, Fortaleza, Brasil, 1995.

[Fid98] C. J. Fidge. *Real-Time Schedulability Tests for Preemptive Multitasking.* Real-Time Systems, 14, pp 61-93, Kluwer Academic Publishers, 1998.

[Fon97] K. V. O. Fonseca. *Uma metodologia de Configuração do Suporte de Comunicação de Sistemas Tempo-Real Críticos.* Tese Dr, LCMI, UFSC, Brasil, 1997.

[fSEC94] International Organization for Standardization/International Eletrotechnical Committee. *Information Technology - Telecommunications and Information Exchange Betweem Systems - Local and Metropolitan Area Networks - Specific requirements- Part 2: Logical Link Control.* ISO/IEC 8802-2, 1994.

[Jún88] L. Nacamura Júnior. *Projeto e implementação de um Núcleo de Sistema Operacional Distribuído com mecanismos para tempo real.* Tese Mestrado, LCMI, UFSC, 1988.

[Kop92] H. Kopetz. *Scheduling.* An Advanced Course on Distributed Systems, Portugal, 1992.

[KS94] K. A. Kettler and J. K. Strosnider. *Scheduling Analysis of the Micro Channel Architecture for Multimedia Applications.* IEEE International Conference on Multimedia and Computing Systems, 1994.

[KSS95a] D. I. Katcher, S. S. Sathaye, and J. K. Strosnider. *Fixed Priority Scheduling with Limited Priority Levels.* To appear: IEEE Transactions on Computers, 1995.

[KSS95b] K. A. Kettler, J. K. Strosnider, and E. J. Snow. *Real-Time Scheduling of Bus Strutures for Multimedia Applications.* Submitted to IEEE Magazine Multimedia, 1995.

[LL73] C. L. Liu and J. W. Layland. *Scheduling algorithms for multiprogramming in hard real-time environment.* Journal of ACM 20(1), pp. 40-61, 1973.

[LR93] G. Le Lann and N. Rivierre. *Real-Time Communications over Broadcast Networks: the CSMA-DCR and the DOD-CSMA-CD Protocols.* Rapoort de Recherche, INRIA, França, 1993.

[MNS96] A. Meschi, M. D. Natale, and M. Spuri. *Earliest Deadline Message Scheduling with Limited Priority Inversion.* Scuola Superiore S. Anna, Pisa, Italy, 1996.

[Oli97] R. S. Oliveira. *Escalonamento de Tarefas Imprecisas em Ambiente Distribuído.* Tese Dr, LCMI, UFSC, 1997.

[SLC95] L.F.G. Soares, G. Lemos, and S. Colcher. *Redes de Computadores: das LANs, MANs e WANs ás redes ATM.* Editora Campus, 2da Edição, 1995.

[Spu96] M. Spuri. *Analysis of Deadline Scheduled Real-Time Systems.* Rapoort de Recherche, INRIA, França, 1996.

[TBW94] K. W. Tindell, A. Burns, and A. J. Wellings. *An Extendible Approach for Analyzing Fixed Priority Hard Real-Time Tasks.* Journal of Real-Time Systems, vol. 6,(Klüwer Academic pub), 1994.

[TBW95] K. Tindell, A. Burns, and A. J. Wellings. *Analysis of Hard Real-Time Communications.* Real-Time Systems. 9. pp.147-171, 1995.